Chapter 1

INTRODUCTION

Kidney diseases are disorders that affect the functions of the kidney**.** During the late stages, kidney diseases can cause kidney failure. According to the data most recently released by the Saudi Center of Organ Transplantation Registry, 10,203 patients diagnosed with kidney disease receive hemodialysis.Diabetes, high blood pressure, and unhealthy lifestyles have led to an increase in the number of patients with CKD. Patients with CKD suffer from various side effects. These complications include damage to the nervous and immune systems that disrupt daily activities. To assist in the prevention of CKD, machine learning techniques can be utilized to diagnose CKD at an early stage.

Artificial Neural Network (ANN) and Support Vector Machine (SVM) are widely used machine learning techniques. Both ANN and SVM have advantages and have been proven to perform excellently in several fields, including medical diagnosis, weather prediction, stock market analysis, and image processing.

SVM performs well and has good accuracy even with limited examples. Due to its fruitful advantages, SVM is distinct from other machine learning techniques ANN has the ability to learn how to perform its functions once it is properly trained and the ability to generalize and produce a reasonable solution to unobserved data.

In this project ANN, RFC ,KNN ,NBC and SVM techniques are applied to a CKD dataset to measure their accuracy and compare their performance.

SVM has an accuracy of 97.5%, KNN has an accuracy of 98% and then RFC has an accuracy of 98.75% which is more or less similar to the accuracy of ANN.

The remainder of this work is organized as follows. Section 2 provides a review of the related literature. Section 3 discusses the proposed machine learning techniques, including the neural network and support vector classifiers. In section 4, we present the empirical studies, including the dataset description, experimental setup, methodology, adopted optimization strategy and parameter search strategy.Section 5 presents the user interface. Section 6 presents the results and discussion, section 7 presents the conclusion.

Chapter 2
LITERATURE REVIEW

Z. Chen, X. Zhang, and Z. Zhang presented the applicability of clinical data and the robustness of three multivariate models in clinical risk assessment of CKD patients [2]. The UCI repository dataset for Chronic Renal Failure (CFR) was used in this study. The three multivariate models used included the K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Soft Independent Modeling of Class Analogy (SIMCA). The authors were able to label the patients as CKD or non-CKD with an accuracy of over than 93%. Another important finding of this paper was that SVM processed the noise disturbance of the composite dataset better and had the highest accuracy in predicting CKD compared to the other two models as its accuracy reached 99%.

In [3], the authors compared three different models for supervised feature selection. In their study, the authors used a supervised rule mining algorithm to explore the feature space and predict CKD stages based on proton Nuclear Magnetic Resonance (1H NMR) data. The first model used by the authors involved global data mining, which is a dataset wide analysis using fisher's exact test in which features with a very small probability (p-value ≤ 0.05) are chosen, and normalized mutual information is used. The second model involved a local approach based on supervised rule mining that the authors used to lift and z-score. The final model consisted of the output of the first model, which served as the input of the second model. After examining the three models, the authors found that the local approach based on the supervised rule mining model was the best model since it had the highest F1 score and mean and the lowest standard deviation.

In [4], the authors developed a decision support system to help doctors predict the occurrence of CRF in patients. This decision was based on various machine learning techniques, including Naive Bayes, LDA, K-Nearest Neighbor (KNN), tree-based decision, and random subspace classification algorithms. These techniques were applied to a UCI repository dataset of CRF.

It was found that the K-Nearest Neighbor (KNN) with random subspace classifier had 94% accuracy in prediction. The authors suggested using kernel or neural based classifiers instead of KNN with a random subspace classifier.

In [5], a C4.5 decision tree algorithm was performed using the UCI repository dataset to diagnose CRF in patients. After applying the algorithm, eight if-then rules were developed to predict CRF. The accuracy of the algorithm was found to be 98.25%. The experiment was performed using 3-fold cross-validation. The researchers also found that serum creatinine, pedal edema, diabetes mellitus, hemoglobin, and specific gravity were the most important features in the dataset. Finally, the authors suggested using a dataset with more diversity in age, gender, and location of patients to allow the algorithm used for diagnosis to be universally applicable.

In [6], the machine learning techniques used for classification included Artificial Neural Network (ANN), Naïve Bayes, and a decision tree.

This system was applied to a dataset from Prince Hamza Hospital in Jordan. The performance of the machine learning techniques was analyzed by calculating the sensitivity, specificity, and accuracy. The authors found that the decision tree was the most accurate technique in predicting CRF. Finally, the authors suggested implementing more machine learning techniques, including Support Vector Machine (SVM), to predict CRF.

Salekin and J. Stankovic introduced two methods in their study to reduce overfitting and identify the most important attributes for CKD prediction. The two methods were LASSO regularization and wrapper. The authors used three different classification methods and compared their accuracy using the F1 measure and root mean square error. The three classifiers included random forest, k-nearest neighbors and neural networks. It was found that the random forest algorithm can detect CKD with the highest accuracy of 0.998 using the F1-measure with a 0.107 root mean square error.

In [7], machine learning techniques, including SVM, decision tree, Naïve Bayes, and KNN, were used to detect whether a patient has CKD using the UCI repository dataset of CRF. The selection of features in the dataset was performed by implementing a ranking algorithm. The algorithm yielded an output of 10 features that were important for CKD prediction. Then, different machine learning techniques were applied to the dataset, and their accuracies were calculated using the root mean square error and mean absolute error and by plotting a receiver operating characteristic curve. The decision tree algorithm was found to be most accurate with an accuracy of 99.75%, while SVM was the second most accurate with an accuracy of 97.75%. Finally, the authors suggested implementing ANN and fuzzy logic to the dataset and analyzing their accuracies.

In [8], the author proposed a new method to automate the diagnosis of CKD by using data mining methods. K-means was applied to the data for preprocessing. The classification methods KNN, SVM, and Naïve Bayes were applied to the preprocessed data as the data mining methods. An accuracy rate of 97.8% was obtained, which was the highest rate, using the proposed method with urine test attributes. The accuracy was increased in the age group of 35 and above using the same attributes. This study also revealed that different combinations of the dataset attributes resulted in different accuracy rates ranging from 83.75% to 97.8%.

In [4], the author suggested using neural based classifiers instead of KNN. This study used ANN as a neural based classifier and achieved a high accuracy of 99.75%. In [6], the authors suggested implementing more machine learning techniques, including Support Vector Machine (SVM). This study utilizes the SVM machine learning technique, reaching performance accuracy of 97.75%. The ANN machine learning technique has been shown to have the highest accuracy compared with the other utilized techniques in all previous studies.

Chapter 3

ALGORITHMS AND ITS IMPLEMENTATION

3.1 Data Preprocessing and Feature Scaling

The dataset has missing values.Missing Data in numerical columns is replaced with Mean.Missing Data in categorical columns is replaced with a global value 'nun'.Categorical values are converted into numerical using the packages LabelEncoder and OneHotEncoder.Encoding is done separately for dependent and independent variables.Data is divided into training set and test set in ratio 4:1.Cross Validation can also be used for splitting training and test set data.Feature Scaling is done using StandardScaler Package.

Code:

```
df = pd.read_csv('Data.csv')
df=df.replace('?',np.nan)
   X =df.iloc[:, :-1].values
   Y = df.iloc[:, 24].values
   #missing values
fromsklearn.preprocessing import Imputer
imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
imputer = imputer.fit(X[:, 0:14])
X[:,0:14] = imputer.transform(X[:,0:14])
   x=pd.DataFrame(X)
   y=pd.DataFrame(Y)
   x=x.replace(np.nan,'nun')
   y=y.replace(np.nan,'nun')
   X=np.array(x)
   Y=np.array(y)
   # Encoding the Independent Variable
fromsklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
```

```
for i in range(14,24):
X[:, i] = labelencoder_X.fit_transform(X[:, i])
   x=pd.DataFrame(X)
   # Encoding the Dependent Variable
labelencoder_y = LabelEncoder()
   Y = labelencoder_y.fit_transform(Y)
   # Splitting the dataset into the Training set and Test set
fromsklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25,
 random_state = 0)
   #Feature Scaling
fromsklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

3.2 Random Forest Classification

Random forest, like its name implies, consists of a large number of individual decision trees .It is a Supervised Machine Learning Algorithm forClassification.Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's predictionIn general, the **more trees in the forest** the more robust the forest looks like. In the same way in the random forest classifier, the **higher the number** of trees in the forest gives **the high accuracy** results.

From ensemble class in sklearn library we can do random forest classification. A class by name RandomForestClassifier is used to do the process of classifying data. The parameters in RandomForestClassifier includes n_estimators,criterion,max_depth, min_samples_split and so on.

3.2.1 Feature Importance in Random Forest:

feature_importances - DataFrame

Index	importance
3	0.211271
11	0.145993
2	0.131649
10	0.117308
7	0.0890827
14	0.0674534
19	0.0450887
6	0.0431979
8	0.0367934
18	0.0173801
15	0.0139444
22	0.0131122
9	0.0117607
12	0.0104461

Figure 1

From the values of feature importance we can infer the weightage or priority of all the features in the dataset .The values range from 0-1 and the features with higher importance have to be given more priority when data is modelled using RFC algorithm.

3.2.2 Implementation of Random Forest

Fit the classifier to the training set and compute the results. Predict the values for the test set according to the model trained. Compute all the evaluation metrics,execution time and time complexity for the model.

Classifier applied to the training data:

from sklearn.ensemble import RandomForestClassifier

```
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',
random_state = 0)
classifier.fit(X_train, y_train)
```

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Parameters:

n_estimators : integer, optional (default=10) -The number of trees in the forest.

criterion : string, optional (default="gini")-The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

random_state : int, RandomState instance or None, optional (default=None)

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by *np.random*.

Classification_report:

	Precision	Recall	F1-score	Support
0	0.97	1.00	0.98	62
1	1.00	0.95	0.97	38
Avg/total	0.98	0.98	0.98	100

3.3 Support Vector Machine

“Support Vector Machine” (SVM) is a supervised machine learning algorithm .In this , 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, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).Support vector can be implemented using SVM class in svm package .The parameters in SVM include C,kernel,degree,gamma and so on.C indicates penalty parameter and kernel specifies type of kernel.Degree represents the degree of the polynomial kernel function. Default is rbf .

3.3.1 Implementation of Support Vector Machine

Fit the classifier to the training data and compute the results.Predict the values for the test set correspondingly.Confusion matrix and accuracy score are computed for the model.

Classifier applied to the training data:

```
from sklearn.svm import SVC

classifier = SVC(kernel = 'sigmoid', random_state = 0)

classifier.fit(X_train, y_train)
```

Support Vector Classification.

The implementation is based on library svm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

kernel : string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices

random_state : int, RandomState instance or None, optional (default=None)

The seed of the pseudo random number generator to use when shuffling the data. If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator.

Classification report for Support Vector Machine

	Precision	Recall	F1-score	Support
0	1.00	0.98	0.99	52
1	0.97	1.00	0.98	28
Avg/total	0.99	0.99	0.99	80

3.4 Naive Bayes Classification

A classification technique based on Bayes' Theorem with an assumption of independence among predictors. It's NAIVE as it assumes that features of a measurement are independent of each other.Bayes theorem-A way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c)

$P(c|x)=P(x|c)P(c)/P(x)$

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

3.4.1 Implementation of Naïve Bayes

Posteriori Probability is based on likelihood measures of data getting into a particular class.After preprocessing the data, apply GaussianNB on training data and predict the classification for test set .

Classifier applied to the training data:

```
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X_train, y_train)
```

Classification_report:

	Precision	Recall	F1-score	Support
0	0.92	0.95	0.94	62
1	0.92	0.87	0.89	38
Avg/total	0.92	0.92	0.92	100

3.5 K-Nearest Neighbours(k-NN)

A classification algorithm that trains data into classes and classifies the data based on distance measures such as ‘Manhatton’ or ‘Euclidean’ or ‘Minkowski’.Distance measures are valid only for continuous variables . For categorical variables hamming distance is used.No. of neighbours K decides the class to which the new data point belongs to.Cross validation can also be used to find out optimal K value .Minkowski with Power parameter 1 acts as **Manhatton**.Minkowski with Power parameter 2 acts as **Euclidean**.Minkowski with Power parameter as any arbitrary value is considered as **Minkowski**.

3.5.1 Implementation of k-NN

After preprocessing of data set, we have to fit the knn to the training data. k =5 and distance measure = minkowski with P=2 are taken.Then classifier is applied on test set to verify if the classification is done right and plotted the points of each class using a unique colour.

Classifier applied to the training data:

```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X_train, y_train)
```

Parameters:

n_neighbors : int, optional (default = 5)

Number of neighbors to use by default for kneighbors() queries.

p : integer, optional (default = 2)

Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

metric : string or callable, default 'minkowski'

the distance metric to use for the tree. The default metric is minkowski, and with p=2 is equivalent to the standard Euclidean metric.

Classification_report:

	Precision	Recall	F1-score	Support
0	1.00	0.97	0.98	62
1	0.95	1.00	0.97	38
Avg/total	0.98	0.98	0.98	100

3.6Multi LayerPerceptron(MLP)

A class of feedforward artificial neural network consisting of at least three layers of nodes: an input layer, a hidden layer and an output layer.A neural network y=f(x;q) is called feedforward as it goes from x to y in forward direction where x is input layer and y is ouput layer.Learning algorithm decides how to use hidden layers to best implement an approximation.They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP

Except for the input nodes, each node is a neuron that uses a nonlinear activation function.MLP utilizes a supervised learning technique called backpropogation for training.

Its multiple layers and non-linear activation like sigmoid or tanh or ReLU or logistic distinguish MLP from a linear perceptron.Solvers such as lbfgs or sgd or adam are used for weights.For output layer we mostly use softmax as activation function .

3.6.1 Implementation of MLP

In MLP, we first fit the preprocessed training data into an MLP.It is done by using object of MLPClassifier, a package for mlp .Then, we find weights of neurons from input layer to first hidden layer and then from first hidden layer to the second hidden layer and so on.Next we find bias values for each hidden layer.For bias, we use “intercepts_” , a list of bias vectors, where the vector at index i represents the bias values added to layer i+1 .Now for the test set , predict the output for all the samples by means of classifier.Form two clusters out of the samples and find out training set score, testing set score and loss function for each iteration of data samples.

Classifier applied to the training data:

```
from sklearn.neural_network import MLPClassifier

clf=MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
    beta_2=0.999, early_stopping=False, epsilon=1e-08,
    hidden_layer_sizes=(5, 2), learning_rate='constant',
    learning_rate_init=0.001, max_iter=200, momentum=0.9,
    nesterovs_momentum=True, power_t=0.5,
    random_state=1, shuffle=True, solver='lbfgs', tol=0.0001,
    validation_fraction=0.1, verbose=False, warm_start=False)
print(clf.fit(X_train,y_train))
```

Multi-layer Perceptron classifier.

This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

Parameters:

hidden_layer_sizes : tuple, length = n_layers - 2, default (100,)

The ith element represents the number of neurons in the ith hidden layer.

activation : {'identity', 'logistic', 'tanh', 'relu'}, default 'relu'

Activation function for the hidden layer.

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

solver : {'lbfgs', 'sgd', 'adam'}, default 'adam'

The solver for weight optimization.

- 'lbfgs' is an optimizer in the family of quasi-Newton methods.
- 'sgd' refers to stochastic gradient descent.
- 'adam' refers to a stochastic gradient-based optimizer

alpha : float, optional, default 0.0001

L2 penalty (regularization term) parameter.

learning_rate : {'constant', 'invscaling', 'adaptive'}, default 'constant' Learning rate schedule for weight updates.

- 'constant' is a constant learning rate given by 'learning_rate_init'.
- 'invscaling' gradually decreases the learning rate learning_rate_ at each time step 't' using an inverse scaling exponent of 'power_t'. effective_learning_rate = learning_rate_init / pow(t, power_t)
- 'adaptive' keeps the learning rate constant to 'learning_rate_init' as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least tol, or fail to increase validation score by at least tol if 'early_stopping' is on, the current learning rate is divided by 5.Only used when solver='sgd'.

max_iter : int, optional, default 200

Maximum number of iterations. The solver iterates until convergence (determined by 'tol') or this number of iterations. For stochastic solvers ('sgd', 'adam'), note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

random_state : int, RandomState instance or None, optional, default None

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by *np.random*.

3.6.2 Backpropogation

Backpropagation algorithms are a family of methods used to efficiently train artificial neural networks (ANNs) following a gradient-based optimization **algorithm** that exploits thechain rule. ... **Backpropagation** requires the derivatives of activation functions to be known at network design time.

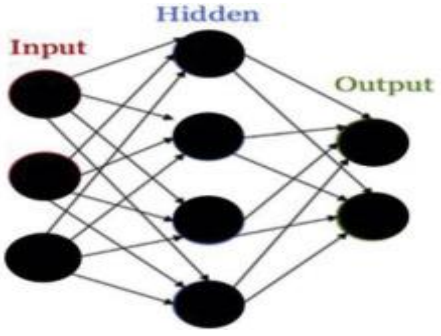

Figure 3-Basic representation of neural network

Time complexity for back propogation is O(n.m.h^k.o.i) where

n –no. of training samples

m- no. of features

k-hidden layers each containing

h-neurons

o-no. of output neurons

i-no. of iterations

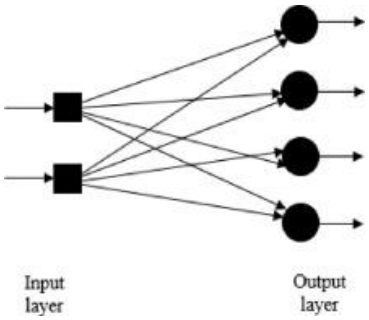

Figure 4-Input and Output Layer

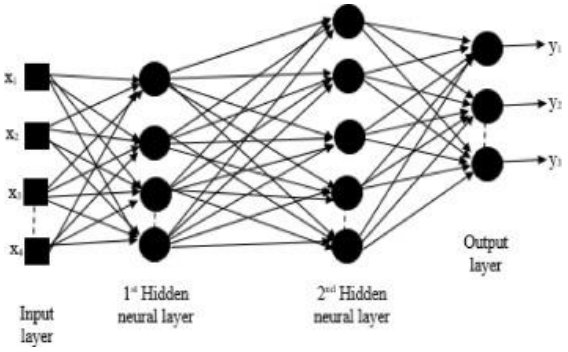

Figure 5-2 Hidden layers represented

Classification_report:

	Precision	Recall	F1-score	Support
0	0.97	1.00	0.98	62
1	1.00	0.95	0.97	38
Avg/total	0.98	0.98	0.98	100

Chapter 4

EMPERICAL STUDIES

4.1 Description of the dataset

This experiment was performed using the CKD dataset available in the UCI machine learning repository. The CKD dataset was re- leased in July 2015 by Apollo Hospitals, Tamilnadu, India. The dataset consists of 24 attributes, including 11 numerical and 13 nominal attributes, collected from 400 instances over a two-month period. These 400 instances consist of 250 CKD patients and 150 non-CKD patients. As a result, this dataset is used for binary classification of either CKD or non-CKD. Table 1 presents a list of the attributes, their types and values.

4.2 Statistical analysis of the dataset

The statistical analysis of the dataset is introduced in Table 2. The mean, median, standard deviation, minimum and maximum of the numerical attributes in the dataset are introduced.The nominal attributes, their labels and their counts are presented in Table 3. The dataset was preprocessed before performing the statistical analysis, and the missing values were replaced with the mean of the numerical distribution.Table 4 presents the correlation coefficients between each numer- ical attribute and the targeted class attribute. Correlation coefficients are used to determine how much each numerical attribute contributes to the target attribute. Correlation coefficients help in the process of feature selection.As shown in Table 4, the largest correlation is approximately 0.40 (blood glucose random attribute). Overall, the correlation between the attributes and the target class is considered weak, and the correlation values ranged from 0.7 to 0.4.

Table 1 : Dataset Description

Attribute	Type	Value
Age	Nominal	Years
Blood Pressure	Nominal	Mm/Hg
Specific Gravity	Nominal	1.005,1.010,1.015,1.020,1.075
Albumin	Nominal	0,1,2,3,4,5
Sugar	Nominal	0,1,2,3,4,5
Red Blood Cells	Nominal	Normal,Abnormal
Pus Cells	Nominal	Normal,Abnormal
Pus Cell Clumps	Nominal	Present,Not Present
Bacteria	Nominal	Present,Not Present
Blood Glucose Random	Numerical	mgs/dl
Blood Urea	Numerical	mgs/dl
Serum Ceratinine	Numerical	mgs/dl
Sodium	Numerical	mEq/L
Pottasium	Numerical	mEq/L
Heamoglobin	Numerical	Gms
Packed Cell Volume	Numerical	--
White Blood Cell Count	Numerical	cells/cumm
Red Blood Cell Count	Numerical	millions/cmm
Hypertension	Nominal	Yes,No
Diabetes Milletus	Nominal	Yes,No
Coronary Artery Disease	Nominal	Yes,No
Appetite	Nominal	Good,Poor
Pedal Edema	Nominal	Yes,No
Anemia	Nominal	Yes,No

The above table represents the data type and value in units of each and every attribute in the dataset .For any health issues like Diabetes,Coronary artery disease ,Anemia are taken as Yes/No type values.RBC and Pus cells are taken values as Normal/Abnormal.

Table 2:Statistical analysis of the dataset

Attribute	Mean	Median	Standard Deviation	Maximum	Minimum
Age	51.483	54	16.975	90	2
Blood Pressure	76.469	80.235	13.476	180	50
Blood Glucose Random	148.037	126	74.783	490	22
Blood Urea	57.426	44	49.286	391	1.5
Serum Creatinine	3.072	1.4	5.617	76	0.4
Sodium	137.529	137.529	9.204	163	4.5
Pottasium	4.627	4.627	2.82	47	2.5
Hemoglobin	12.526	12.526	2.716	17.8	3.1
Packed Cell Volume	38.884	38.884	8.151	54	9
White Blood Cell Count	8406.12	8406.12	2523.22	26400	2200
Red Blood Cell Count	4.707	4.707	0.84	8	2.1

The values of mean,median ,standard deviation,maximum and minimum of each and every attribute are computed from data values in the data set and results are entered in the above table .

Table 3:Analysis of Nominal Attributes

Attribute	Label	Count
Specific Gravity	1.005	7
	1.01	84
	1.015	75
	1.02	153
	1.025	81
Albumin	0	245
	1	44
	2	43
	3	43
	4	24
	5	1
Sugar	0	339
	1	13
	2	18
	3	14
	4	13
	5	3
Red Blood Cells	Normal	353
	Abnormal	47
Pus Cells	Normal	324
	Abnormal	76
Pus Cell Clumps	Present	42
	Not present	358
Bacteria	Present	22
	Not present	378
Hypertension	Yes	147
	No	253
Diabetes Milletus	Yes	137
	No	263
Coronary Artery Disease	Yes	34
	No	366
Appetite	Good	318
	Poor	82
Pedal Edema	Yes	76
	No	324
Anemia	Yes	60
	No	340

Table 4:Correlation Coefficient between each attribute and target attribute

Attribute	Target Attribute	Correlation Coefficient
Age	Class	0.22541
Blood Pressure	Class	0.2906
Blood Glucose Random	Class	0.40137
Blood Urea	Class	0.37023
Sodium Creatinine	Class	0.29408
Sodium	Class	-0.3423
Pottasium	Class	0.07692
Hemoglobin	Class	-0.7296
Packed Cell Volume	Class	-0.6901
White Blood Cell Count	Class	0.20527
Red Blood Cell Count	Class	-0.5909

Attribute having highest positive correlation coefficient have more impact on the occurrence of CKD and if the attribute with least correlation coefficient has a high value ,then the patient has least chance of CKD.From the table we can infer that Heamoglobin has least value which means the person with more haemoglobin has very less chance of CKD and Blood Glucose has highest value which means the person having less glucose levels are more intended to have CKD problem.

Chapter 5
User Interface Creation

The User Inerface was created using python tkinter.After the input is given the user can choose any algorithm by clicking the button.The respective algorithm will run in the background and the output will be displayed.

The code for the button is:

```
window=tk.Toplevel()
window.title("Result")
window.geometry = ("1000000, 1000000")
output = sa(buttons())
outputnum = tk.Label(window,height=5,width=250,text= str(output[0]))
outputnum.pack()
outputnum1 =tk.Label(window,height=5,width=25,text=str(output[1]))
outputnum1.pack()
outputnum2 =tk.Label(window,height=5,width=50,text=str(output[2]))
outputnum2.pack()
button4=tk.Button(window,text="close",height=5,width=25,command=window.destroy)
button4.place()
button4.pack()
```

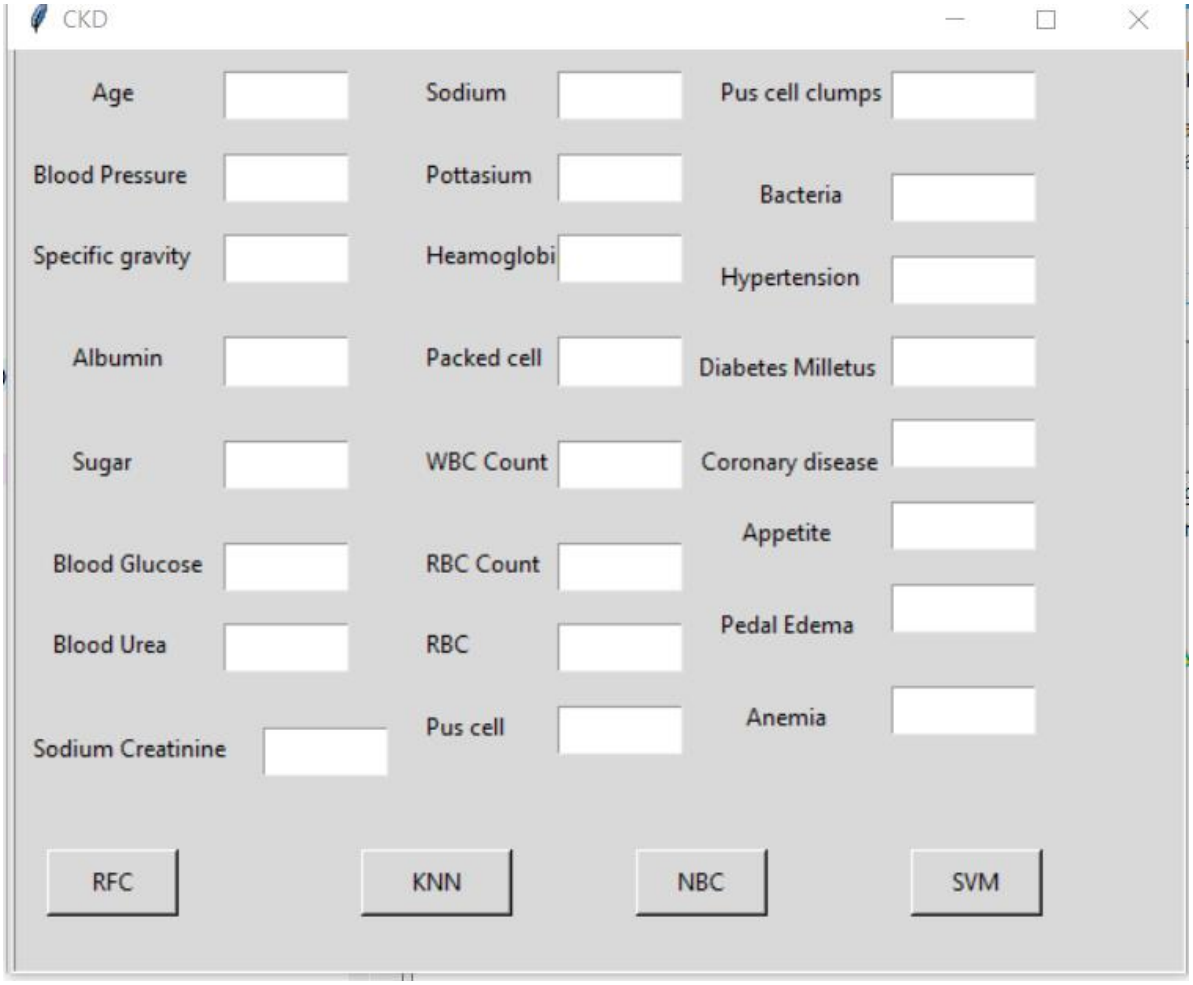

Figure 6 GUI for Input

The values of the features obtained from the medical test should be entered here.The user can choose the required algorithm and click the respective button.By clicking the button the algorithm runs by taking the values as input and predicts the result.

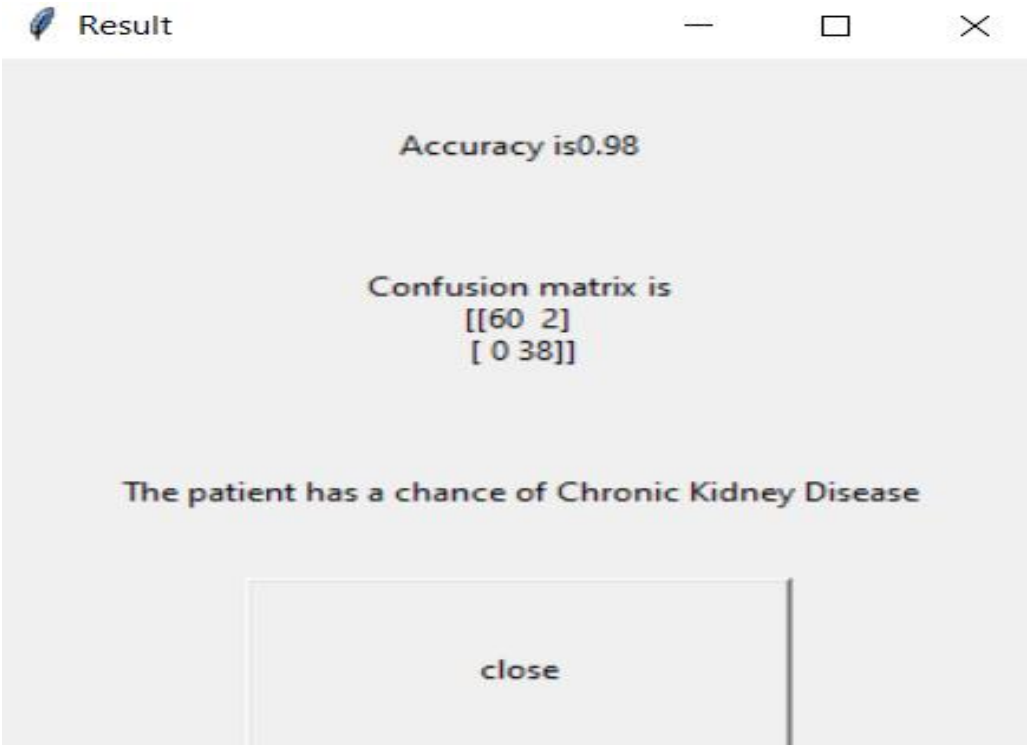

Figure 7 GUI for Output

The output frame will be as shown above.The accuracy and confusion matrix of the algorithm are also displayed along with the result.

Chapter 6
RESULTS AND DISCUSSIONS

Confusion Matrix for all the models:

RFC:

	Predicted CKD	Predicted Not
Actual CKD	62	0
Actual Not CKD	2	36

SVM:

	Predicted CKD	Predicted Not
Actual CKD	51	1
Actual Not CKD	1	27

KNN:

	Predicted CKD	Predicted Not
Actual CKD	60	2
Actual Not CKD	0	30

Naive Bayes:

	Predicted CKD	Predicted Not
Actual CKD	59	3
Actual Not CKD	5	33

MLP:

	Predicted CKD	Predicted Not
Actual CKD	60	0
Actual Not CKD	2	36

Algorithm	Accuracy	Recall	Precision	F1 Score
Random Forest Classification	98	0.98	0.98	0.98
K Nearest NeighborsClassification	98	0.98	0.98	0.98
Support Vector Classification	97.5	0.99	0.99	0.99
Naïve Bayes Classification	92	0.92	0.92	0.92
Multi Layer Perceptron	98	0.98	0.98	0.98

1)Support Vector Classification

- Time Complexity : **O(nSVd)** where nSV is the number of support vectors and d is no. of input dimensions
- Execution time : **0.02254 sec**

2)Random Forest Classification

- Time Complexity : **O(v * n log(n))**, where n is the number of records and v is the number of variables/attributes
- Execution time : **0.01958 sec**

3)Naïve Bayes Classification

- Execution time : **0.1163 sec**
- Time complexity : **O(Nd**) for d features

4)K nearest Neighbors Classification

- Execution time : **0.0494 sec**
- Time complexity : **O(logN)** –average **O(kN^(1-1/k))**-worst

5)Multilayer perceptron (MLP)

- Execution time : **0.1596 sec**
- Time complexity : **O(n^2)**

Chapter 7

CONCLUSION

Various Machine Learning techniques used in this project have advantages and have been proven to perform excellently in several fields. ANN has been proposed to better predict CKD and has been compared against other algorithms, which has exhibited the highest accuracy in previous studies. The dataset was first preprocessed, and the numerical missing values were replaced by the mean and categorical missing values were replaced by a global value. Cross-validation was used as the algorithm for the partitioning of the training and test datasets with the ratio (75: 25). It was found that ANN performs better with an accuracy of 98% using the optimized features and can be used with large datasets and the accuracy wouldn't decrease. The chances of a person having a CKD can be effectively found out using Support Vector Classification,Random Forest Classification,Naïve Bayes Classification,K nearest Neighbors Classification,Multilayer perceptron (MLP). These can be used in the medical field to help doctors identify quickly if a person has a chance of Chronic Kidney Disease.

On application of algorithms the accuracies obtained for Random Forest ,Support Vector, Naive Bayes , K-Nearest Neighbours and MLP are 0.9875,0.975,0.92,0.98 and 0.98 respectively. Random Forest has the highest performance and best results amongst all as feature importance is considered a key role in that algorithm. For a larger scale of data, MLP works well as it has less complexity and it also overcomes the problems of overfitting and underfitting by means of activation functions and optimization techniques are implemented using solvers .Blood Sugar levels, Glucose levels and Heamoglobin play the role of key features having more importance compared to others.

Chapter 8

REFERENCES

[1] A. a Al-Sayyari, F. a. Shaheen,"End-stage chronic kidney disease in Saudi Arabia. A rapidly changing scene",Saudi Med. J., 32 (2011), pp. 339-346.

[2] Z. Chen, X. Zhang, Z. Zhang,"Clinical risk assessment of patients with chronic kidney disease by using clinical data and multivariate models"Int. Urol. Nephrol., 48 (12) (2016), pp. 2069-2075,*http://doi.org/10.1007/s11255-016-1346-4.*

[3]M.M. Luck, A. Yartseva, G. Bertho, E. Thervet, P. Beaune, N. Pallet,"Metabolic profiling of H-1 NMR spectra in Chronic Kidney Disease with local predictive modelling",2015,IEEE 14th International Conference on Machine Learning and Applications (Icmla) (2015), pp. 176-181,*http://doi.org/10.1109/icmla.2015.155*.

[4] A.M. Al-Taee, A.Y. Al-Hyari, M.A. Al-Taee,"Clinical decision support system for diagnosis and management of chronic renal failure",2013, IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT) (2013).

[5]A.A. Serpen,"Diagnosis rule extraction from patient data for chronic kidney disease using machine learning",Int. J. Biomed. Clin. Eng., 5 (2) (2016), pp. 64-72, *http://doi.org/10.4018/IJBCE.2016070105.*

[6]R. Ani, G. Sasi, R.S. U, O.S. Deepa,"Decision support system for diagnosis and prediction of chronic renal failure using random subspace classification",2016 ,International Conference on Advances in Computing, Communications and Informatics(ICACCI) (2016),pp. 12871292,*http://doi.org/10.1109/ICACCI.2016.7732224.*

[7]N. Tazin, S.A. Sabab, M.T. Chowdhury,"Diagnosis of Chronic Kidney Disease Using Effective Classification and Feature Selection Technique",(2013), *http://doi.org/10.1109/MEDITEC.2016.7835365.*

[8]S.B. Akben,"Early stage chronic kidney disease diagnosis by applying data mining methods to urinalysis , blood analysis and disease history",IRBM, 39 (5) (2018), pp. 353-358,*https://doi.org/10.1016/j.irbm.2018.09.004.*

[9] H. Polat, H. DanaeiMehr, A. Cetin,"Diagnosis of chronic kidney disease based on support vector machine by feature selection methods",J. Med. Syst., 41 (4) (2017), p. 55,*http://doi.org/10.1007/s10916-017-0703-x.*

[10] A. Salekin, J. Stankovic,"Detection of Chronic Kidney Disease and Selecting Important Predictive Attributes",(2016),*http://doi.org/10.1109/ICHI.2016.36.*

[11]M. Bhattacharya, C. Jurkovitz, H. Shatkay,"Chronic Kidney Disease Stratification Using Office Visit Records : Handling Data Imbalance via Hierarchical Meta-Classification" vol. 18 (2018).

[12]R.G. Brereton, G.R. Lloyd,"Support vector machines for classification and regression"Analyst, 135 (2) (2010), pp. 230-267,*http://doi.org/10.1039/B918972F.*

[13] "SurveyImportance of artificial neural networks for location of faults in transmission systems",Surveyor (2017), pp. 357-362.

[14]T.S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, D. Haussle,"Supp ort vector machine classification and validation of cancer tissue samples using microarray expression data",16 (10) (2000), pp. 906-914.

[15]A.A.A.A. Adewumi, T.O. Owolabi, I.O.I.O. Alade, S.O.S.O. Olatunji,"Estimation of physical, mechanical and hydrological properties of permeable concrete using computational intelligence approach",Appl. Soft Comput., 42 (2016), pp. 342-350,*http://doi.org/10.1016/j.asoc.2016.02.009.*