Problem statement

- Identify the factors causing chronic kidney disease
- Build a model that can help to determine if a patient is suffering from kidney chronic disease or not

Exploratory Data Analysis

- Dataset contains 25 columns, including the label/class/target column.
- Label column has two values – one specifying the presence of disease and the other not. So, it is a classification problem (and not regression).
- There are 400 samples – with 250 samples for positive disease and 150 with the disease label as negative.
- Of the remaining 24 feature columns, 14 are nominal and 11 are numerical columns.
- There are missing values in all the features with the missing values % ranging from as high as 38% in rbc (red blood cells) to as low as 0.25% in ane (anemia).
- Missing values are represented with '?'.
- All the columns are with data type 'object'.
- Nominal columns - sg(specific gravity), al(albumin) and su(sugar) have numerical values, but is not mentioned, if they are ordinal.
- Not all numerical features follow normal distribution. Some of them are skewed – eg. bgr, sc, bu, sod.
- Just looking at the data and with the primitive knowledge on the domain, we could expect to see a correlation between
  - the numerical feature rbcc (red blood cell count) and the categorical features rbc (red blood cells) and ane (anemia).
  - the numerical feature bp (blood pressure) and the categorical feature htn(hypertension)
  - numerical feature bgr (blood glucose random) and categorical feature su (sugar)
- There could be other correlated features too.

Data pre-processing

- There are white spaces in the values of the feature columns. Trim the white spaces (required in identifying duplicates, during encoding, etc.,)
- See if there are any duplicate samples and if there are, remove them. While there is merit in retaining the duplicates to reflect the real world situation, in order to reduce the bias, we shall remove duplicates in this study.(actually there were no duplicates!!)
- Convert the feature columns from 'object' datatype to numerical/categorical based on the provided dataset information.
- Replace the missing values '?' with the numpy NaN.
- Since the missing values are relatively high in proportion in most of the columns to the number of samples (eg. 38% missing values in rbc, 26.5% in wbcc, 22 % in pot, etc.,) and we only have 400 samples, it might not be a good idea to drop the samples/features with missing values.
- We shall fill the missing values in the numerical features with the median and with the most frequently occurring value (mode) for the categorical features.
- Find the correlation between the features and draw a heatmap. From the heatmap we could see that hemo and pcv are strongly correlated(0.9), hemo and rbcc are correlated(0.8), pcv and rbcc are correlated(0.69). This means that hemo is an important feature. We will get to ascertain this later when we visual the model based on decision tree classifier.
- We shall encode the categorical features and covert them into numerical values. We could go for label encoding or one hot encoding. Since, we do not know if the provided nominal values are ordinal in nature, label encoding might not be the right choice as it would assign weight to the otherwise unordered values. Hence, we shall choose one hot encoding.

Data pre-processing – continued

- Missing values % (top 7 columns) and the correlation of features

	Missing Values	% of Total Values
rbc	152	38.00
rbcc	131	32.75
wbcc	106	26.50
pot	88	22.00
sod	87	21.75
pcv	71	17.75
pc	65	16.25

	age	bp	bgr	bu	sc	sod	pot	hemo	pcv	wbcc	rbcc
age	1	0.136316	0.230731	0.194291	0.133985	-0.0860401	0.0493991	-0.178308	-0.212796	0.0937945	-0.203199
bp	0.136316	1	0.15018	0.180841	0.143184	-0.100705	0.0636671	-0.279303	-0.289237	0.0221768	-0.219317
bgr	0.230731	0.15018	1	0.118859	0.0688864	-0.130569	0.0527317	-0.254435	-0.2582	0.119881	-0.214094
bu	0.194291	0.180841	0.118859	1	0.581517	-0.308806	0.339771	-0.541635	-0.523091	0.0383623	-0.465243
sc	0.133985	0.143184	0.0688864	0.581517	1	-0.624036	0.204751	-0.342492	-0.338611	-0.0103334	-0.321737
sod	-0.0860401	-0.100705	-0.130569	-0.308806	-0.624036	1	0.0696122	0.331483	0.343555	0.00800996	0.313929
pot	0.0493991	0.0636671	0.0527317	0.339771	0.204751	0.0696122	1	-0.0964281	-0.117795	-0.0755627	-0.118226
hemo	-0.178308	-0.279303	-0.254435	-0.541635	-0.342492	0.331483	-0.0964281	1	0.84749	-0.137978	0.667512
pcv	-0.212796	-0.289237	-0.2582	-0.523091	-0.338611	0.343555	-0.117795	0.84749	1	-0.175226	0.693473
wbcc	0.0937945	0.0221768	0.119881	0.0383623	-0.0103334	0.00800996	-0.0755627	-0.137978	-0.175226	1	-0.153776
rbcc	-0.203199	-0.219317	-0.214094	-0.465243	-0.321737	0.313929	-0.118226	0.667512	0.693473	-0.153776	1

From the heatmap we could see that hemo and pcv are strongly correlated(0.9), hemo and rbcc are correlated(0.8), pcv and rbcc are correlated(0.69). This means that hemo is an important feature. Note: We are not removing the features in the study.

Data pre-processing - continued

- We shall perform outlier analysis by calculating the z-score. Let us remove the outliers where the z score >= 3. It is a conscious decision, as the models trained without the removal of outliers provided accuracy of 100% which is a clear case of overfitting. Hence, even though the samples are lesser, let us remove the outliers with the above mentioned threshold for the Z-score. We ended up removing 116 outlier samples.
- As mentioned in the EDA, some of the numerical features do not follow normal distribution and are skewed. Let us normalize them by performing log transformation.
- We need to scale the features, so that the values of the different features are comparable. Standardization (subtracting mean and scaling variance) is required for many methods like regularized linear regression, K-NearestNeighbors. We could use standard scaler, min max scaler or robust scaler. Let us go with standard scaler.

Modelling – Training and predicting

- As it goes with saying, nearly 70% of the time should need to be spent on understanding the data and pre-processing them. Having done that to the best of our efforts, let us now create models based on different linear and non linear classification algorithms, evaluate them, compare them and finalise a model.
- Let us split the sample data into training data with 80% of the samples and the remaining 20% to predict.
- We could do the cross validation with XGBoost and find the important features. We will repeat this action of identifying important features with other classifiers as we go along. Here, we see that sc (serum creatinine) is the most important feature in determining the presence of the kidney disease. This is aligned with what we read in https://www.kidney.org/atoz/content/understanding-your-lab-values where it says
  - ***Serum Creatinine:*** *Creatinine is a waste product in your blood that comes from muscle activity. It is normally removed from your blood by your kidneys, but when kidney function slows down, the creatinine level rises.*
- Let us define a function which will take the inputs as classifier name, training and test dataset and perform the fit and predict functions with those classifiers.
- We shall then call this function with the following classifiers
  - **Logistic Classification:** Since the outcome is binary and we have a reasonable number of examples at our disposal compared to number of features, this approach seems suitable. At the core of this method is a logistic or sigmoid function that quantifies the difference between each prediction and its corresponding true value.
  - **Liner Discriminant Analysis:** Logistic regression is intended for two-class or binary classification problems. It can be extended for multi-class classification, but is rarely used for this purpose. Unstable With Well Separated Classes. Logistic regression can become unstable when the classes are well separated. Unstable With Few Examples. Logistic regression can become unstable when there are few examples from which to estimate the parameters. Linear Discriminant Analysis does address each of these points and is the go-to linear method for multi-class classification problems. Even with binary-classification problems, it is a good idea to try both logistic regression and linear discriminant analysis.

Modelling – Training and predicting

- **Support Vector Machine:** SVM aims to find an optimal hyperplane that separates the data into different classes.
- **Decision Tree:** Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.
- **k Nearest Neighbors:** It uses the k closest neighbors of a data point to determine which class should that data point belong to.
- **Naive Bayes:** Naive Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. Naive Bayes classifiers work well in many real-world situations such as document classification and spam filtering.
- **Ada Boosting:** Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. AdaBoost was the first really successful boosting algorithm developed for binary classification. It focuses on classification problems and aims to convert a set of weak classifiers into a strong one.
- **Gradient Boosting:** Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (eg. decision trees).
- **Random Forest Classifier:** It comes under the category of ensemble methods. It employs 'bagging' method to draw a random subset from the data, and train a Decision Tree on that.
- **Extra Tree Classifier:** Extra trea algorithm is quite similar to Random Forest - but splits are selected on random instead of using some criterions.

- While we need not have to model with the all the above classifiers since we know our data and can use the algorithms that better suit our data, let us model with all the classifiers as this assignment is for learning.

https://towardsdatascience.com/

Modelling – Training and predicting

- Please find below, the training and the test accuracy with different classifiers along with their F-scores

	Classsifer	Training_Accuracy	Test_Accuracy	Training F1 score	Test_F1 score
0	LogisticRegression	1.000000	0.982456	1.000000	0.971223
1	LinearDiscriminantAnalysis	0.977974	0.947368	0.961905	0.912409
2	SVC	0.986784	0.982456	0.977230	0.971223
3	DecisionTreeClassifier	1.000000	0.982456	1.000000	0.992908
4	KNeighborsClassifier	0.977974	0.947368	0.961905	0.912409
5	GaussianNB	0.982379	0.982456	0.969582	0.971223
6	AdaBoostClassifier	1.000000	0.982456	1.000000	0.971223
7	GradientBoostingClassifier	1.000000	0.982456	1.000000	0.971223
8	ExtraTreesClassifier	1.000000	0.982456	1.000000	0.971223
9	RandomForestClassifier	0.995595	0.982456	0.992439	0.971223

- Since most of the models are with the accuracy in around 98%, we suspect overfitting. However, we only have 400 samples. Remember we removed 114 outliers from 400 samples. When modelled without removing any outliers, accuracy were 100% in most of the models.
- Let us do a comparative study of different classification algorithms and choose one that fits our dataset relatively better

Modelling – Choosing a model

- **Logistic regression:** no distribution requirement, perform well with few categories categorical variables, compute the logistic distribution, good for few categories variables, easy to interpret, compute CI, suffer multicollinearity
  - Since our dataset matches the expectation in terms of distribution and categorical features, we can consider this algorithm for our problem statement.
- **Linear discriminant analysis:** require normal, not good for few categories variables, compute the addition of Multivariate distribution, compute CI, suffer multicollinearity
  - Expects normal distribution of data and that is not always the case with the features in our dataset. So, let us not go with this classifier.
- **SVM:** no distribution requirement, compute hinge loss, flexible selection of kernels for nonlinear correlation, not suffer multicollinearity, hard to interpret
  - We could use this classifier, but hard to interpret.
- **Decision Trees:** no distribution requirement, heuristic, good for few categories variables, not suffer multicollinearity (by choosing one of them)
  - Since our dataset matches the expectation in terms of distribution and categorical features, and overcomes the multicollinearity issue that we might have introduced due to one hot encoding of the categorical features (column count increased twice to 50)
- **KNN:** Features need not be linearly separable. Well suited for multimodal. Takes up lot of memory. Works well for small number of dimensions.
  - Since our dataset is small, memory is generally not a constraint in our problem. There are more dimensions due to one hot encoding. We can choose to consider this classifier.
- **Gaussian NB:** generally no requirements, good for few categories variables, compute the multiplication of independent distributions, suffer multicollinearity
  - We may not be able to use this classifier, since this assumes independence between features, which is not the case in our problem.
- **Bagging, boosting, ensemble methods(RF, Ada, Gradient Boosting):** generally outperform single algorithm listed above.
  - Since this is generally aggregating the trees and we have seen that decision tree can better suit our needs, we shall go with one of the ensemble methods based classifier. **We shall go with Random Forest.**

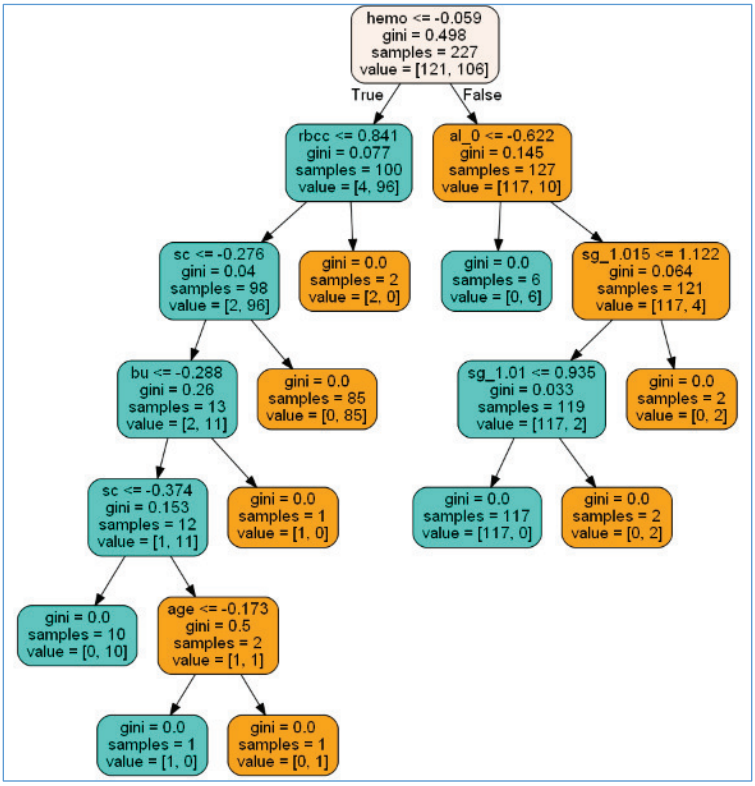
	Classifer	Training_Accuracy	Test_Accuracy	Training F1 score	Test_F1 score
0	LogisticRegression	1.000000	0.982456	1.000000	0.971223
1	LinearDiscriminantAnalysis	0.977974	0.947368	0.961905	0.912409
2	SVC	0.986784	0.982456	0.977230	0.971223
3	DecisionTreeClassifier	1.000000	0.982456	1.000000	0.992908
4	KNeighborsClassifier	0.977974	0.947368	0.961905	0.912409
5	GaussianNB	0.982379	0.982456	0.969582	0.971223
6	AdaBoostClassifier	1.000000	0.982456	1.000000	0.971223
7	GradientBoostingClassifier	1.000000	0.982456	1.000000	0.971223
8	ExtraTreesClassifier	1.000000	0.982456	1.000000	0.971223
9	RandomForestClassifier	0.995595	0.982456	0.992439	0.971223

We shall choose Random Forest classifier

https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms

Decision tree visualisation

- Here is the visualisation of the decision tree. As we expected during the pre-processing phase, we could see that hemo (Hemoglobin) is the important feature.



Evaluation of models

- Cross validation (CV) is one of the technique used to test the effectiveness of the models. It is primarily based on re-sampling procedures to evaluate a model. To perform CV we need to keep aside a sample/portion of the data on which is do not use to train the model, later us this sample for testing/validating. There are many methods for CV like Train-test split, K fold, K fold repeated, Leave one, Stratified.
- We have used K fold repeated method. This is where the k-fold cross-validation procedure is repeated n times, where importantly, the data sample is shuffled prior to each repetition, which results in a different split of the sample.
- Decision tree classifier seem to be a winner here with the lesser variance.

	Classsifer	cv_mean	cv_variance	cv_standardDeviation
0	LogisticRegression	0.987325	0.000322	0.017942
1	LinearDiscriminantAnalysis	0.951053	0.000763	0.027625
2	SVC	0.982030	0.000266	0.016319
3	DecisionTreeClassifier	0.957763	0.000706	0.026570
4	KNeighborsClassifier	0.959875	0.000718	0.026801
5	GaussianNB	0.982406	0.000236	0.015367
6	AdaBoostClassifier	0.988716	0.000197	0.014019
7	GradientBoostingClassifier	0.977149	0.000594	0.024363
8	ExtraTreesClassifier	0.982738	0.000427	0.020662
9	RandomForestClassifier	0.971817	0.000544	0.023329

ROC Curve

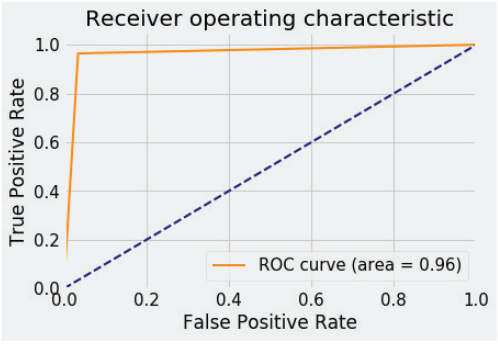
- An additional metric called as Receiver Operator Characteristics(ROC) curve will be used. It plots the curve of True Positive Rate vs the False positive Rate, with a greater area under the curve indicating a better True Positive Rate for the same False Positive Rate. This can be helpful in this case as simply knowing the number of correct predictions may not suffice.
- Following are the ROC scores for the different classifiers
  - For classifier LogisticRegression, ROC score is 1.000000
  - For classifier LinearDiscriminantAnalysis, ROC score is 0.982143
  - For classifier SVC, ROC score is 0.982143
  - For classifier DecisionTreeClassifier, ROC score is 1.000000
  - For classifier KNeighborsClassifier, ROC score is 0.982143
  - For classifier GaussianNB, ROC score is 0.982143
  - For classifier AdaBoostClassifier, ROC score is 1.000000
  - For classifier GradientBoostingClassifier, ROC score is 1.000000
  - For classifier ExtraTreesClassifier, ROC score is 1.000000
  - For classifier RandomForestClassifier, ROC score is 1.000000
- We could not conclude much from the scores!!

Fine tuning the models

- We used the grid search method to see if we can improve the performance of the models with different values for the given hyper parameters.
- Let us fine tune the models – Random Forest, SVC and KNN.
- We have shown here the results of fine tuning the Random Forest model. Results of other models can be seen in the Jupyter notebook.
- We can see the optimizing the model brings the accuracy to a relatively better accuracy of 96.5% and the ROC score of 96%.

```
For classifier RandomForestClassifier:
Unoptimized model
------
Accuracy score on test data: 0.9825
F-score on test data: 0.9712

Optimized Model
------
Final accuracy score on the test data: 0.9649
Final F-score on the test data: 0.9643
Best parameters:
{'max_depth': 3, 'max_features': 'auto', 'n_estimators': 15, 'oob_score': False}
For classifier RandomForestClassifier, ROC score is 0.964901
```



Modelling with the top 10 important features

- From a medical perspective, it may be important to know which features are most influential in determining whether a person has a disease or not. Also, in future, it may be possible that more examples are collected for this dataset, and we may want to use only the k best features for this dataset for classification purpose. Keeping this in mind, we will use the Random Forest Classifier to extract the features of greatest importance.
- As we had seen before hemo (Heamoglobin), sc (Serum Creatinine) seems to emerge as the important features among others.
- When trained the model with these top important features, we could see that, when compared to the original non tuned model, the accuracy reduces by 5.3% and F score by 4.59% and when compared to the fine tuned model, the accuracy reduces by 3.64% and the F-score by 1.79%.

```
top 10 important features are Index(['pcv', 'sc', 'hemo', 'dm_yes', 'rbcc', 'htn_no', 'bp', 'sg_1.015',
       'age', 'al_1'],
      dtype='object')


Final Model trained on full data
------
Accuracy on test data: 0.9825
F-score on test data: 0.9926

Final Model trained on reduced data
------
Accuracy on test data: 0.9298
F-score on test data: 0.9470
```

Conclusion

- Random Forest classifier seems to be giving, relatively the realistic accuracy among the other models. All the other classifiers seem to be overfitting. Partly, it is also because of the small size of the dataset.

- Random forest classifier:
  - Final Model trained on full data – before fine tuning
    - Accuracy on test data: 0.9825
    - F-score on test data: 0.9926
  - Final Model trained on full data – after fine tuning
    - Final accuracy score on the test data: 0.9474
    - Final F-score on the test data: 0.9124
  - Final Model trained on reduced data
    - Accuracy on test data: 0.9298
    - F-score on test data: 0.9470
  - Cross validation
    - Number of folds: 5, Repeat: 10
    - Accuracy Mean: 0.971817
    - Variance: 0.000544
    - Std.Dev: 0.023329

Random forest classifier:

Final Model trained on full data – before fine tuning

> Accuracy on test data: 0.9825
> F-score on test data: 0.9926

Final Model trained on full data – after fine tuning

> Final accuracy score on the test data: 0.9474
> Final F-score on the test data: 0.9124

Final Model trained on reduced data

> Accuracy on test data: 0.9298
> F-score on test data: 0.9470

Cross validation

> Number of folds: 5, Repeat: 10
> Accuracy Mean: 0.971817
> Variance: 0.000544
> Std.Dev: 0.023329