Chronic Kidney Disease Analysis

Group Members: Chaitanya, Anupam, Krishh and Abhinav

Problem Statement

- Chronic Kidney disease is progressive loss in kidney function over a period of months or years. If kidney stops to function properly this may lead to severe consequences and Patients may develop HighBP, anaemia etc other severe ailments.

- Chronic Kidney disease can be prevented and easily managed if caught in early stages , it is often the case that it goes unchecked until the disease has progressed to more advanced stages.

- In this analysis, we are trying to analyse the Chronic Kidney Disease data gathered for 2 months from a hospital, to determine if a patient is suffering from Kidney Chronic Disease, so that we can train a Machine learning model to predict occurrence of such disease well in advance.

Members Details

- Chaitanya (2018ab04519)
- Ramakrishnan kk(2018ab04666)
- Abhinav Kumar(2018ab04668)
- Anupam (2018ab04646)

Table Of Contents

The Data Story

- We are provided with data in csv format and we have taken help of Python pandas library to read the data first and convert it into dataframe.

Data Collection & Preliminary Analysis

Reading the dataset given with the problem, and converting into dataframe.

In [2]:
```
df = pd.read_csv('kidneyChronic.csv')
```

In [3]:
```
# columns
df.columns
```

Out[3]:
```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
       'sc', 'sod', 'pot', 'hemo', 'pcv', 'wbcc', 'rbcc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'class'],
      dtype='object')
```

Understanding the Data

```
In [25]: # columns
         df.columns

Out[25]: Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
                'sc', 'sod', 'pot', 'hemo', 'pcv', 'wbcc', 'rbcc', 'htn', 'dm', 'cad',
                'appet', 'pe', 'ane', 'class'],
               dtype='object')
```

bp - blood pressure
sg - specific gravity
al - albumin
su - sugar
rbc - red blood cells
pc - pus cell
pcc - pus cell clumps
ba - bacteria
bgr - blood glucose random
bu - blood urea
ane - anemia

sc - serum creatinine
sod - sodium
pot - potassium
hemo - hemoglobin
pcv - packed cell volume
wc - white blood cell count
rc - red blood cell count
htn - hypertension
dm - diabetes mellitus
cad - coronary artery disease
appet - appetite
pe - pedal edema

In [33]:
```
# Getting Familier with the dataset
df.head(10)
```

Out[33]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wbcc	rbcc	htn	dm	cad	appet	pe
0	48	80	1.02	1	0	?	normal	notpresent	notpresent	121	...	44	7800	5.2	yes	yes	no	good	no
1	7	50	1.02	4	0	?	normal	notpresent	notpresent	?	...	38	6000	?	no	no	no	good	no
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	...	31	7500	?	no	yes	no	poor	no
3	48	70	1.005	4	0	normal	abnormal	present	notpresent	117	...	32	6700	3.9	yes	no	no	poor	yes
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	...	35	7300	4.6	no	no	no	good	no
5	60	90	1.015	3	0	?	?	notpresent	notpresent	74	...	39	7800	4.4	yes	yes	no	good	yes
6	68	70	1.01	0	0	?	normal	notpresent	notpresent	100	...	36	?	?	no	no	no	good	no
7	24	?	1.015	2	4	normal	abnormal	notpresent	notpresent	410	...	44	6900	5	no	yes	no	good	yes
8	52	100	1.015	3	0	normal	abnormal	present	notpresent	138	...	33	9600	4	yes	yes	no	good	no
9	53	90	1.02	2	0	abnormal	abnormal	present	notpresent	70	...	29	12100	3.7	yes	yes	no	poor	no

10 rows × 25 columns

- We can clearly see above data is a combination of **nominal** as well as **numerical** data.
- Missing values seems to be replaced by ‘?’ already, which we have handled in our analysis gracefully.

- Data Types of all the columns is of object type.

- We don’t have a missing values because its been replaced by ‘?’.

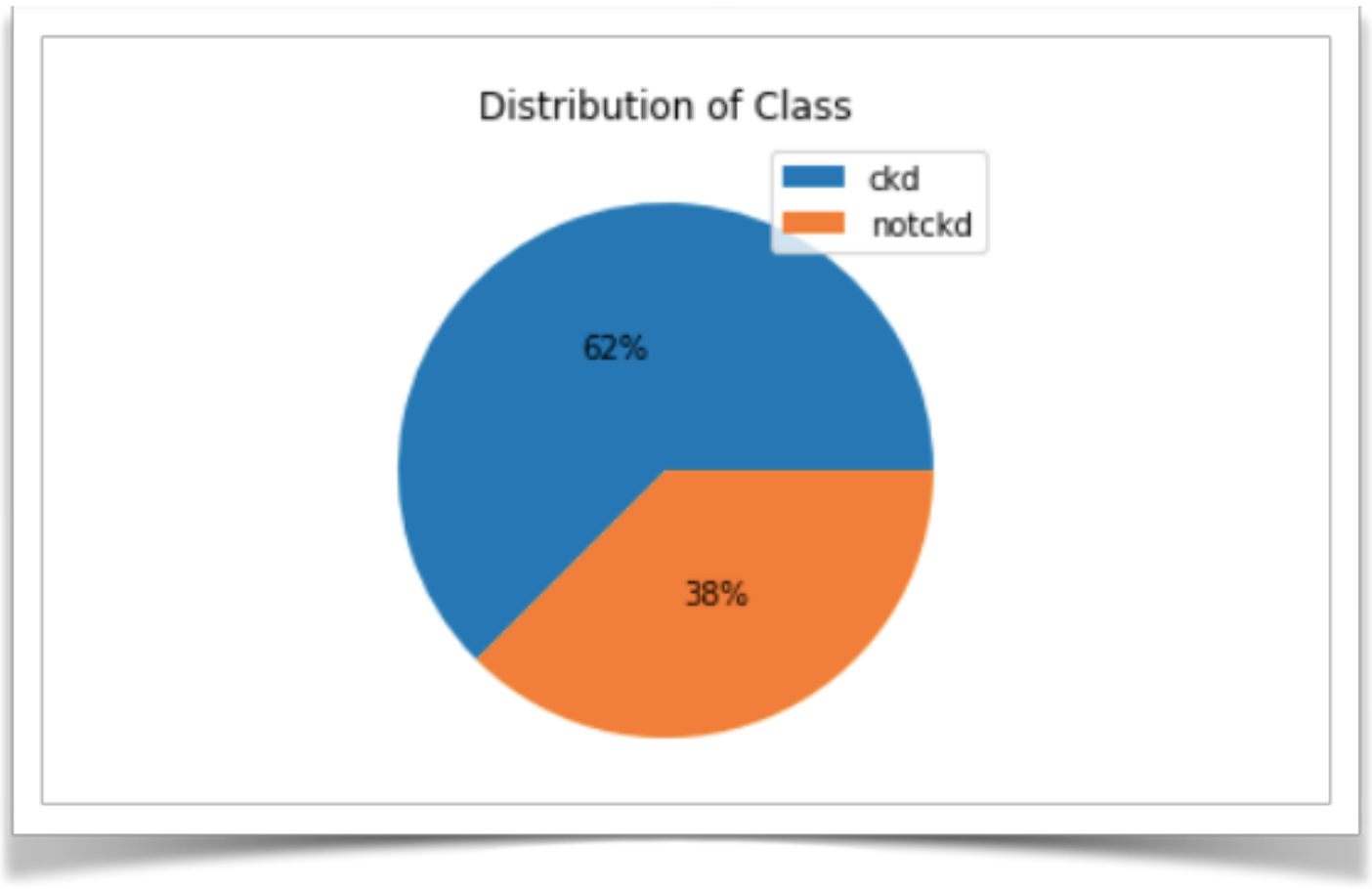
- Class label counts:

```
In [5]: df["class"].value_counts()

Out[5]: ckd        250
        notckd     150
        Name: class, dtype: int64
```

```
In [6]: df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 400 entries, 0 to 399
        Data columns (total 25 columns):
        age      400 non-null object
        bp       400 non-null object
        sg       400 non-null object
        al       400 non-null object
        su       400 non-null object
        rbc      400 non-null object
        pc       400 non-null object
        pcc      400 non-null object
        ba       400 non-null object
        bgr      400 non-null object
        bu       400 non-null object
        sc       400 non-null object
        sod      400 non-null object
        pot      400 non-null object
        hemo     400 non-null object
        pcv      400 non-null object
        wbcc     400 non-null object
        rbcc     400 non-null object
        htn      400 non-null object
        dm       400 non-null object
        cad      400 non-null object
        appet    400 non-null object
        pe       400 non-null object
        ane      400 non-null object
        class    400 non-null object
        dtypes: object(25)
        memory usage: 78.2+ KB
```



Pie Chart to describe the distribution of records labeled with ckd (Chronic Kidney Disease) and non cod

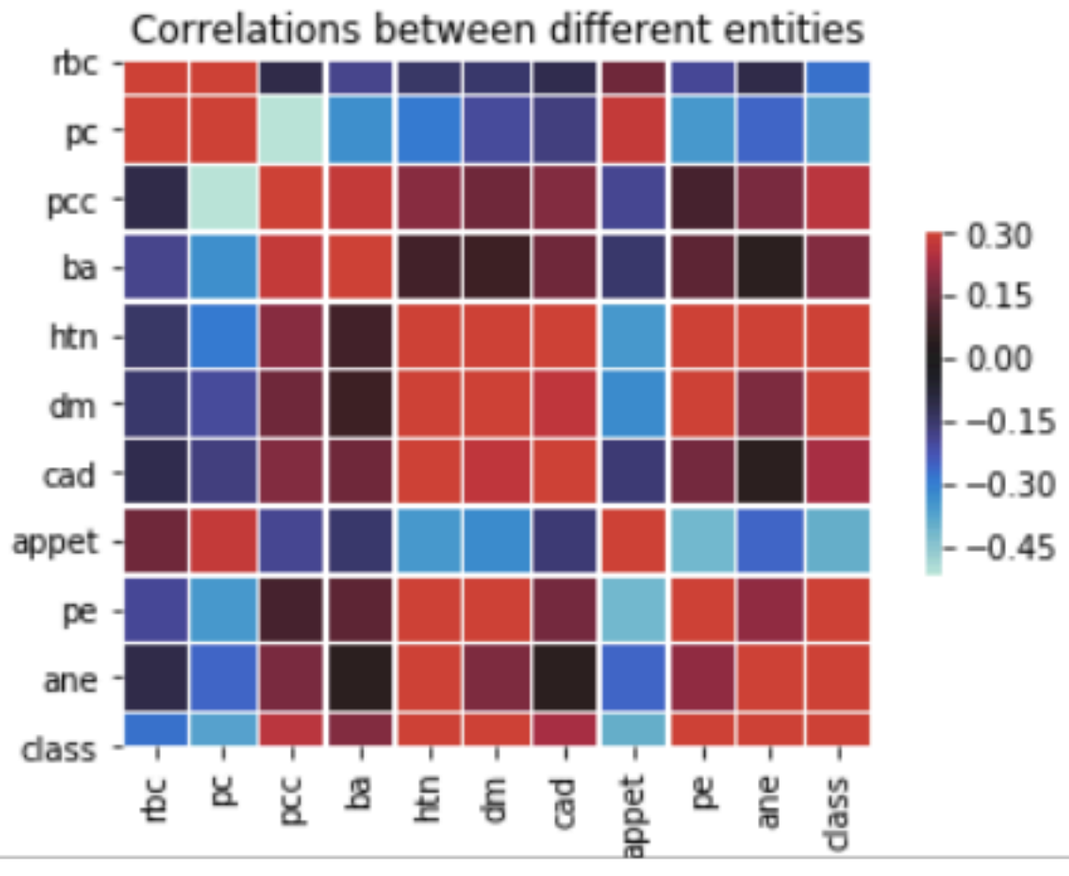
Preprocessing Techniques used

- Since our data contains both numerical as well as nominal data , so we need to come out with a technique to encode the nominal data to do mathematical operations. There are lots of techniques available to do the same but we went through simple approach of “**hot encoding**” because the data is not that complex and its easier to apply Hot encoding on simple data.

- **Missing Values:**
  We have used median as a measure to replace missing values for numerical columns and high frequency values for the nominal columns to replace missing values.

- Heat Map for feature reduction:

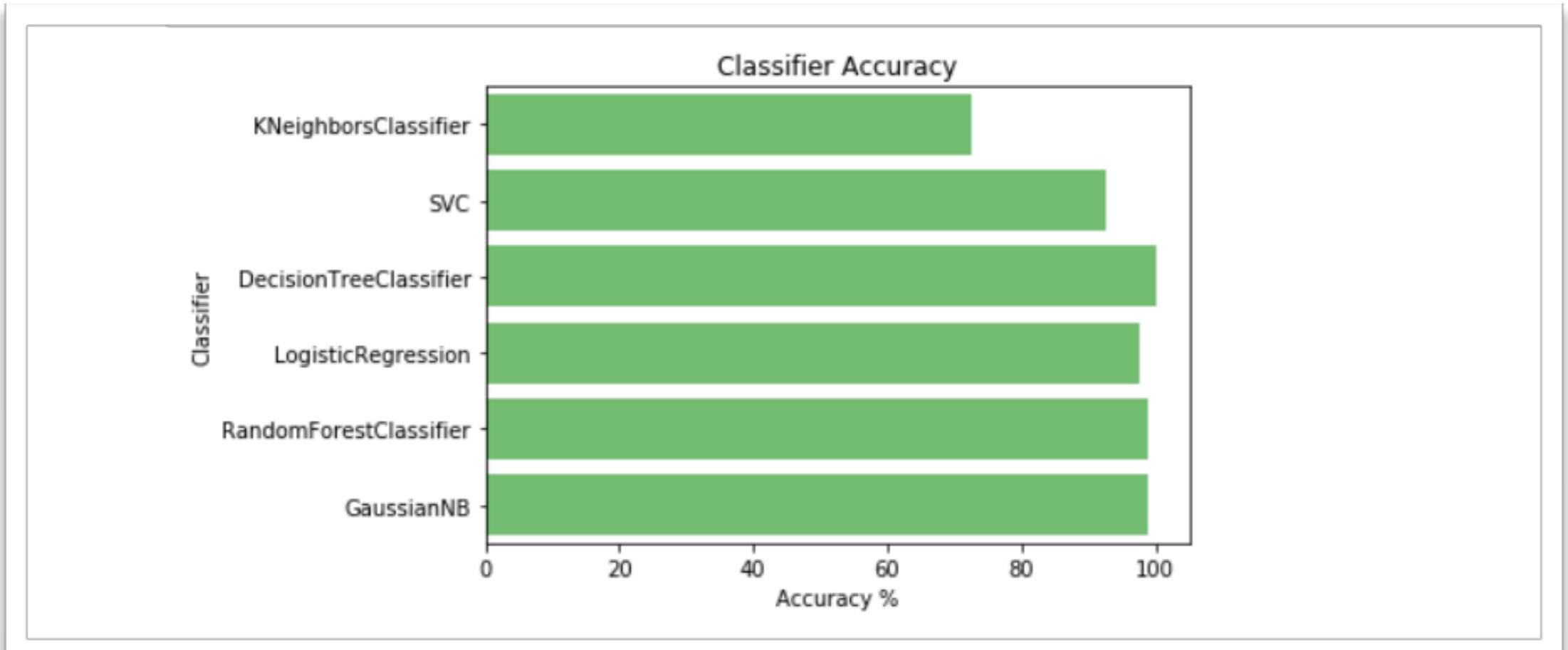
In [15]:
```
corr_df = filled_data_median.corr()
sns.heatmap(corr_df, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.title('Correlations between different entities')
plt.show()
```



Modelling

- K Nearest Neighbours
- Support Vector Machine
- Decision Tree /Random Forest
- Logistic Regression
- Naive Bayes

Comparison of different Models



Conclusion

- We have tried to predict if an individual has Chronic Kidney Disease based on data gathered. This dataset had 400 instances with different labels including a class which indicated if the individual had CDK. We have performed a lot of data cleaning activities including replacing missing values with values based on an algorithm chosen by us. We trained and tested our model and we have gotten very good prediction rates.

- In conclusion we got the desired result and therefore totally fulfilled our goal. It could also be added that our work group was very happy about the results, since we have managed to achieve such high accuracies. We have also learned various techniques in both preparing the data and using it for training various models.

- As it can be clearly seen from the Accuracy measure of Different models, Decision Tree is having best accuracy score. But we might have to try and test it against more diverse and complex set as well.

!! Thank You !!