MINIMIZING PENDING CASES IN INDIAN COURTS

USING MACHINE LEARNING TECHNIQUES

Adarsha M, Ankita S, Nilutpal G, Pallavi S, Sumathi PS, Padmaja K and Narayana D

Great Learning

Abstract

How can artificial intelligence (AI) impact the practice of law? In a general parlance, legal practice which is thought to require advanced cognitive abilities should have little bearing on AI barring significant technical advances. In an environment of legal and factual uncertainty, attorneys regularly combine abstract reasoning and problem solving skills. Although, significant developments have been made, the AI algorithms still lacks the ability to replicate most human intellectual abilities. It is still falling short in advanced cognitive processes that are basic to legal practice. Given these for a fact, a general impression would be that we are yet to see a world where computers can replicate the human order cognition that is displayed by the trained attorneys and thus AI will have little impact to the abstract and uncertain field of law.

Although a broader conclusion rules out the usage of AI in the practice of law, it does miss out on few tasks for which AI can still provide a reasonable conclusion. When considered outside this domain, AI techniques have been successful in applying tasks that were earlier considered to require human intelligence, e.g. language translation. The results of these are not perfect always, however any instances where an approximation is acceptable AI can possibly help in a great way to reduce human dependency and its intelligence driven decision making. This body of work is an attempt to look at opportunities in the practice of law which can be assisted by AI techniques and technologies despite their limitations relative to human cognition.

This study focuses on a class of AI methods known as "machine learning" techniques and their potential impact upon legal practice. This work in no way to be considered as a suggestion that all or most of the tasks routinely performed by attorneys are automatable given the current state of AI technology.

Keywords: Legal Case Outcome, Machine Learning, Classification Techniques, Artificial Intelligence

1. Introduction

The initial sections of this article will focus on providing a background of the Indian legal system, its subjective limitations and our approach towards collection of data on which the Machine Learning algorithms will be applied. The later sections will investigate the basic machine learning algorithms and their implementations. This section will also highlight the significant limitations of these methods. We conclude our work with the

justifications as well as presenting a high level business value in the AI implementation in legal practice.

1.1. Indian judicial system

The Indian judicial system has been a subject of debate across all sections of citizenries for its slowness and delay in tending justice. The system needs some serious and sincere ways to bring legal cases to their conclusion in a more efficient and timely manner. Estimates from multiple sources suggest that there are more than 30 million cases pending in various courts across the country. Most cases drag on for several years, without reaching a conclusion. Shortage of judges and lawyers is suggested to be one of the reasons for such a delay.

In several cases, under trials seem to be spending more time in jail than the period of punishment for their crime if convicted by court. Litigants routinely miss court dates, causing hearings to get repeatedly postponed. A work around to this could be to use technology to make the judicial process more streamlined.

The current government has emphasized the need to involve technology and communication to streamline the judicial process. And recent court appearances via video conferences are in some way showing adaptation of technology. This would not only save precious time, effort, and money; but also help concluding cases at a better pace.

2. Problem statement & objectives study

The subject we picked up is sensitive and has been a constant concern across India. The legal system which at times gets touted as being paralyzed or handicapped needs to see some benefit from the advancements in data analytics. The amount of cases that are currently waiting for a hearing is significant and the same is pending owing to the lack of resources and able infrastructure.

The prime objective of this study is to develop a prediction system which can help legal firm augment their advice to clients about the case's success rate. This intends to help the respective clients decide whether their legal case is worthy enough for a trial in court or can be absolved outside the court, thereby saving time and money.

Using Machine Learning Algorithms, we can identify random patterns and insights which will give the legal practitioners an upper edge over their competitors and help in the prediction of the probable outcome of a case.

This study could be a prototype of a greater legal adviser application. Hence, the development has been primarily concentrated only for certain sections of Indian Penal Code (IPC).

- In scope – Case Outcome Prediction is being developed with only the IPC section 420. The study is being driven only from an academic intent.
- Out of Scope – All IPC sections other than 420 are not being considered towards this project.

3. Literature survey

Predicting legal case outcome is one of the interesting problems to be handled both from legal domain and machine learning. As mentioned earlier, if we could predict legal case outcome with good accuracy, lawyer can save their time, clients' time & money and courts' time. This leads to lesser pending cases in courts & faster judgments. In [1], Authors have used supervised learning approach to predict US Supreme Court decisions & got precision accuracy close to 70%. Authors in [2] also got accuracy close to 70%. In one of the references in

[5] which could not verified, BBC reported 79% accuracy. One of the Canadian startup in [6] claims above 90% accuracy but further details are missing. In other references also, different startups claim accuracy close to 90%.

The novel methodologies for data extraction and prediction beat all these claims by a significant margin. Our accuracy stands at 98% which is above 8% from the latest market standards**.**

4. Data collection

The data to be collected was a significant challenge as the domain (LEGAL) is not technology affluent (at least in India). Although data from other countries were available, we felt that it may not make any significance in the legal parlance of India.

Our dataset contains subset of annotated metadata for "Cheating and dishonestly inducing delivery of property" related cases heard by Courts since 2015. The primary sources for our data are the judgement documents across these courts. We went through more than 100 case documents to come up with 58 features that signify a legal case.

The list of features identified is as follows:

Nature of Disposal	Case Type	Court Number
Court Name	Judge	Judge Gender
Judgement Date	Total Number of Sections	Section 1
Section 2	Section 3	Section 4
Section 5	Section 6	Section 7
Section 8	Section 9	Section 10
FIR Number/Year	Police station	Investigating officer
Case Number	Year	Complainant
Total Accused	Accused #	Accused Name
Accused Gender	Accused Age	Accused Confessed? (plea)
Date Of first Hearing	Complainant advocate	Prosecution advocate
Advocate Defendant	Number of Prosecution witnesses	Names of prosecution witnesses
PW's Examined?	Number of hostile witnesses	Defence witnesses
Charge sheet	Points for consideration	Exhibits on behalf of prosecution P series
Number of exhibits considered	Exhibits on behalf of court Cseries	Exhibits on behalf of accused Dseries
Total Number of Material Objects	Charges proved	Charges not proved
Issues Proved	Issues Not Proved	Accused released on bail
Accused committed to prison	Sentence of Imprisonment granted	Fine with Imprisonment (Rs)
Term Served in Prison(days)	Set off (if any)	Judgement
Citations		

5. Data processing

The 58 key information fields were then compared across all cases. After analysing the fields, it was noticed that the data is not consistent for most of the cases and made us to drop quite a few of these features. A final list of 15 features was obtained that are consistent across all judgement documents and made relative sense in predicting the judgement of the case. The final dataset consists of 202 observations spread across 15 key features. The key features retained are as follows:

Sl #	**Feature Name**	**Description**	**Data type**	**Value**
1	**ipc_420**	Binary indicator to confirm if the case is filed under IPC 420	Categorical	Yes=1, No=0
2	**ipc_120b**	Binary indicator to confirm if the case is filed under IPC 120b	Categorical	Yes=1, No=0

3	**ipc_471**	Binary indicator to confirm if the case is filed under IPC 471	Categorical	Yes=1, No=0
4	**ipc_468**	Binary indicator to confirm if the case is filed under IPC 468	Categorical	Yes=1, No=0
5	**ipc_34**	Binary indicator to confirm if the case is filed under IPC 34	Categorical	Yes=1, No=0
6	**jud_gender**	Gender of the judge presiding over the case	Categorical	Male=0, Female=1
7	**jud_date**	Date when judgement was meted	Date	Date
8	**tot_sec**	Total number of sections filed for the case	Numeric	Number
9	**case_no**	Unique number of the case	Categorical	Multiple Factors
10	**Comp**	Complainant name *	String	Name
11	**tot_accu**	Total number of accused presented in the case	Numeric	Number
12	**accu_gender**	Gender of the individual accused	Categorical	Male=0, Female=1
13	**accu_no**	Sequence number of the accused	Categorical	Multiple Factors
14	**accu_age**	Age of the accused	Numeric	Number
15	**Judgement**	Judgement given in the case	Categorical	Guilty=1, Not guilty=0

Additionally, there was some data imputation as well performed for few of the fields.
Accused Age (accu_age) was found to have two outliers, 0 and Dead. Both values were replaced by 39 which is the approximate mean of the other values.

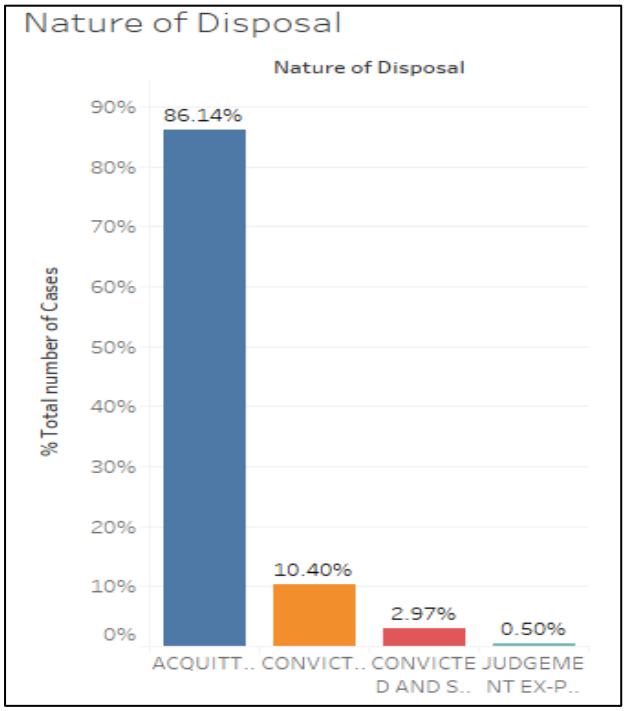
6. Exploratory Data Analysis

6.1 General Overview of Data

Record Count-

- Total number of observations = 202.
- Total number of features = 58 (Additional feature of filename will be omitted from the final workout)

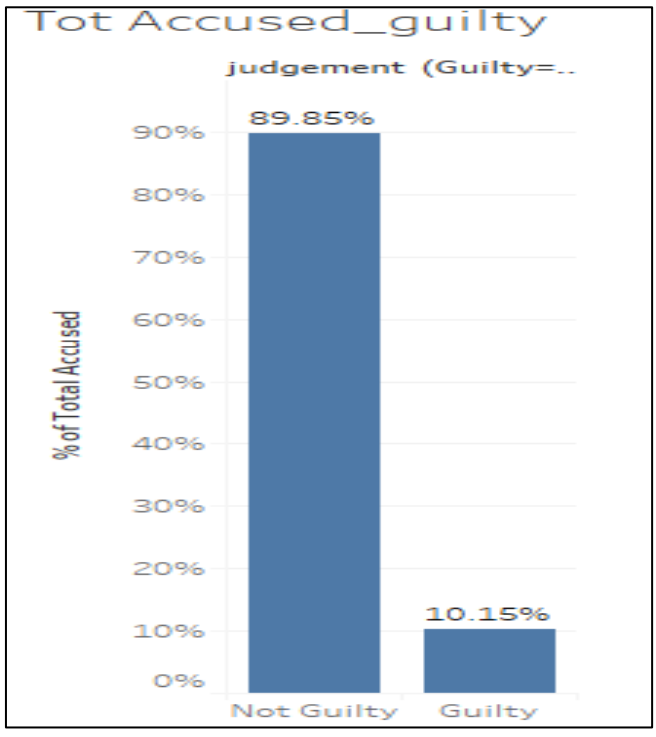
Proportion of cases based on Nature of Dismissal -



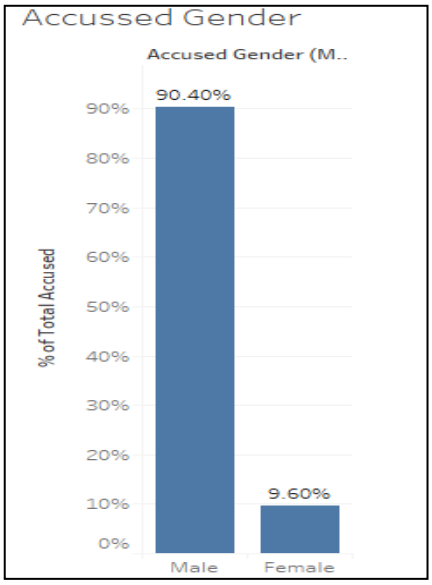
As we can notice, the majority of the cases referred were acquitted (86%) against 13% convicted.

Proportion of Guilty vs Not -Guilty -

Data showed about 90% being not guilty against 10% being guilty.

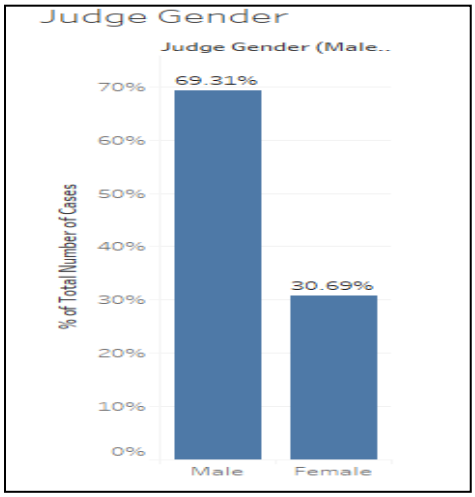


Proportion of Accused gender -



Out of the 10% found guilty, majority of them were male (90.40%). Females were only 9.6%.

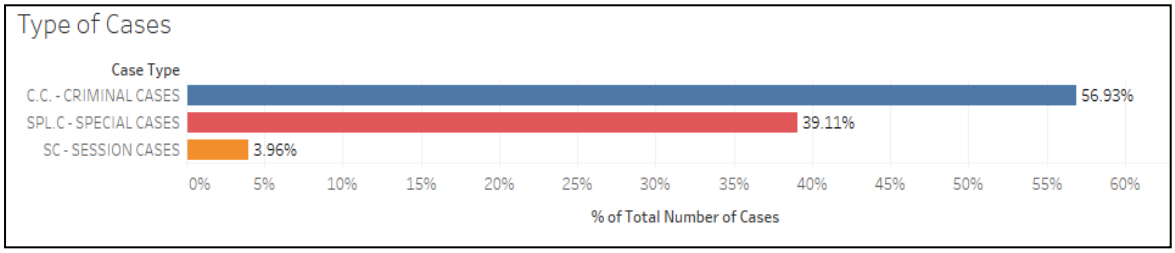
Proportion of Judge gender -



The observations had ~69% male judges against ~31 % of female judges.

Type of cases -

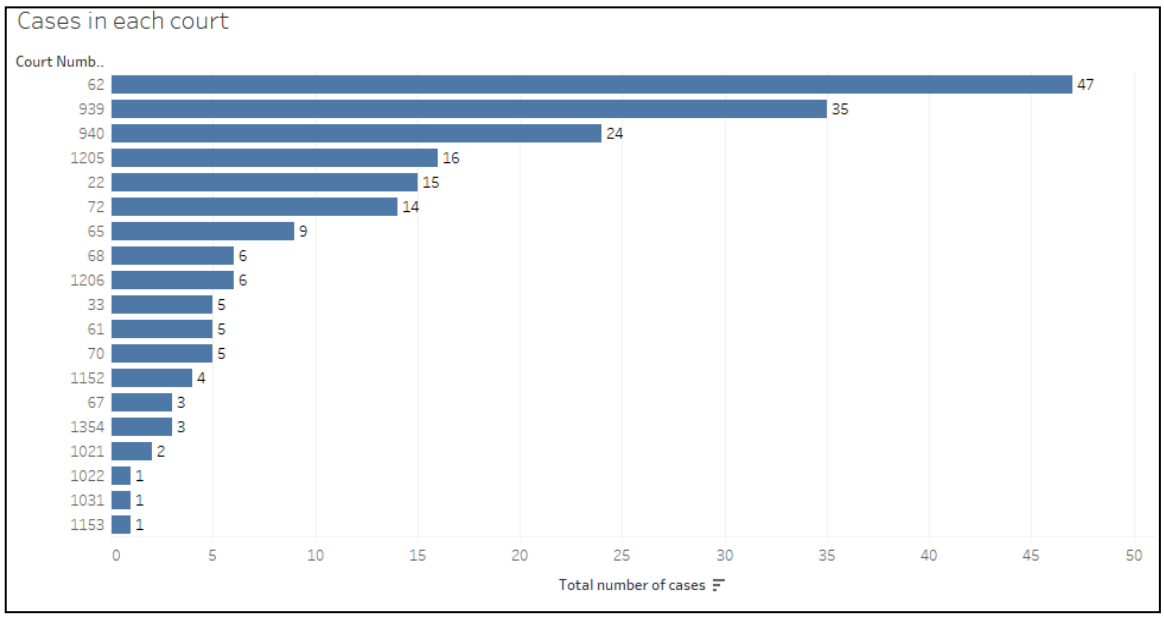
The majority of the cases were criminal cases ~67% followed by special cases filed (~40%) by Crime Branch, CBI, etc.



* Criminal Cases – The cases involving criminal act against a society or section of its population.
* Sessions Cases – All cases punishable with imprisonment for life or with imprisonment for a term exceeding seven years (Code of Criminal Procedure).
* Special Cases – Some special interest cases involving a fast track judiciary or focused cases involving central agencies. Not a conclusive existing criterion available to separate these.

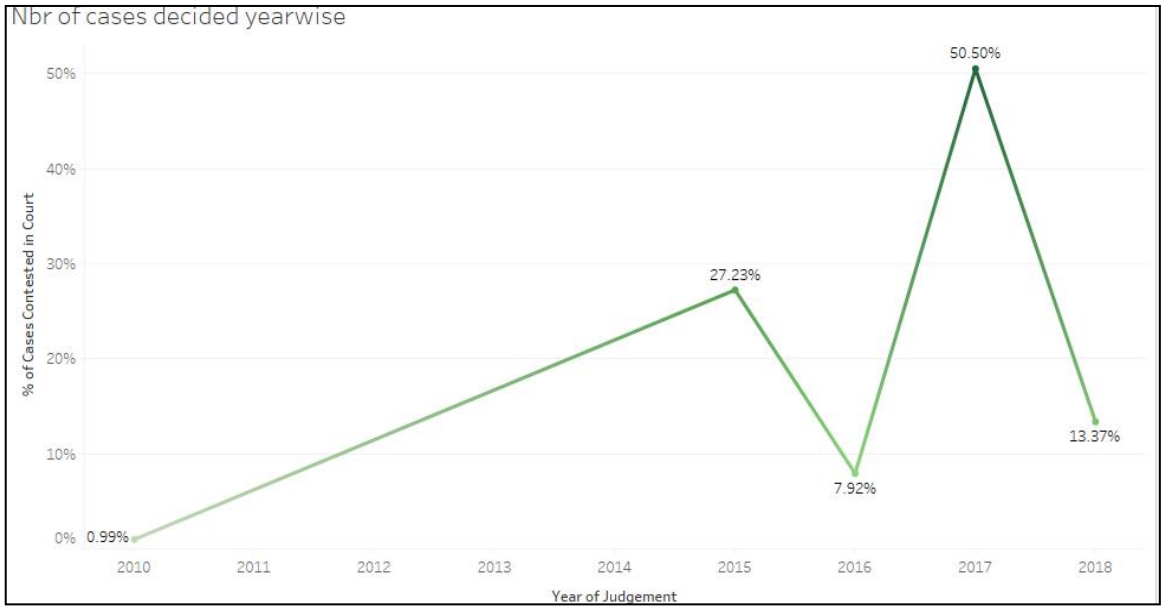
Courts vs number of cases -

The court number 62 has been noticed to have handled majority of the cases with 47, followed closely by court number 939 which handled 35 cases.



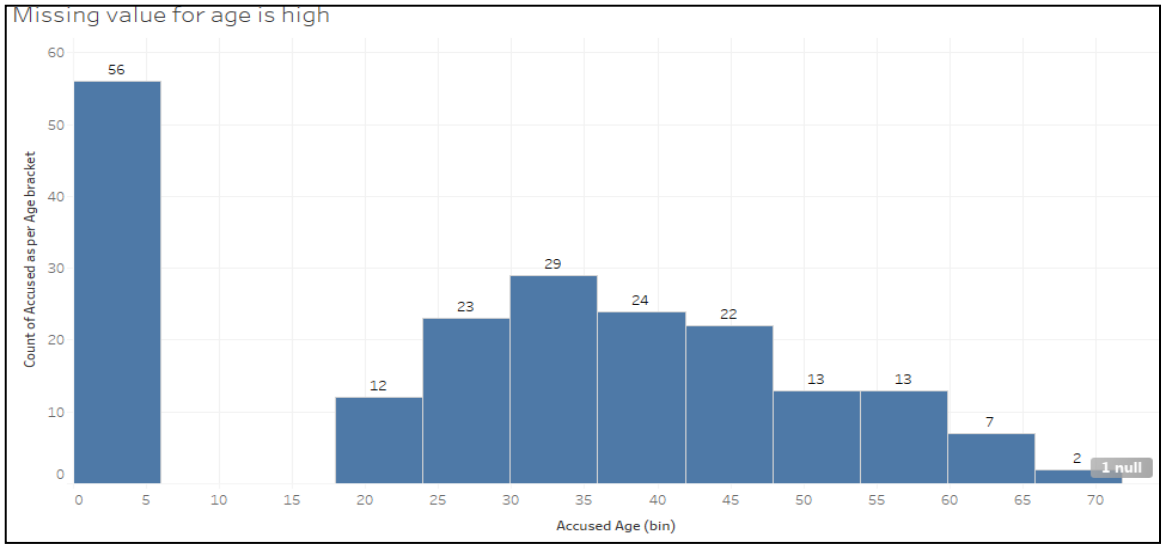
Cases by year of sentencing -

From the period of search from 2010 to 2017. The ~61% of the cases were sentenced in 2017 with 2018 addressing ~13 % of cases.



Missing values -

The analysis by missing values noticed that fairly all features had some missing entries. And the age is one of the features with the maximum amount of missing entries of 56.
Also, overall the age of accused spanned from 20 to 70 years.



7. Standard algorithms legal case outcome prediction

Based on the exploratory data analysis, we want to use the following set of algorithms. Each algorithm has its own merits and demerits. In the following table, we list standard algorithms for expected judgement outcome along with their advantages and disadvantages.

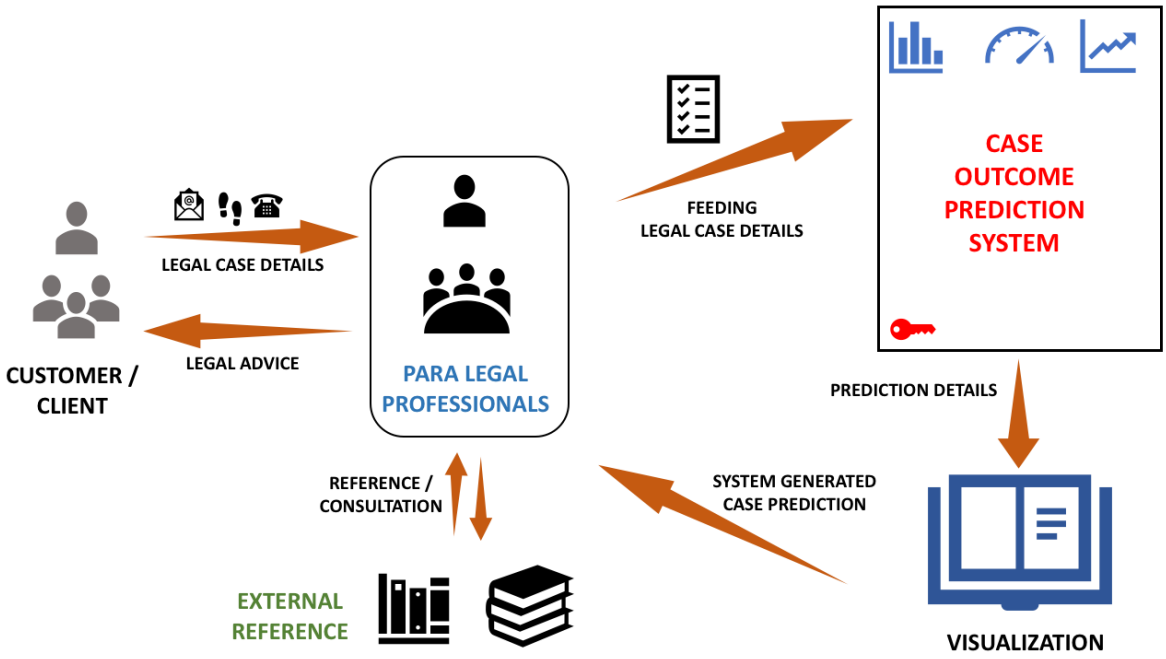
Techniques	Advantages	Disadvantages
Logistic regression	• **Easy to interpret.** • **Logistic regression outputs a number between 0 and 1, which can loosely be interpreted as a probability** • **The feature weights easily indicate which features are more important in determining the classifications than others.**	• Is prone to overfitting. Logistic regression may require a fairly large training set before it can make accurate predictions outside of the training set. • Not an online technique. Incorporating new data requires running gradient descent again. • Has trade-offs with gradient descent. The faster gradient descent determines the final feature weights, the more likely it is to miss the optimal weights. It can sometimes be difficult to figure this out the optimal speed versus accuracy trade-off, and trial-and-error can be time consuming
Naïve Bayes	• **Performs as well as much more complex algorithms with an upper hand in ease of implementation.** • **Simple to interpret. Each feature has a probability, which can tell the most strongly associated with certain classifications.** • **An online technique, that supports incremental training. To include one new data sample, we can update the probabilities- and there is no need to go through the original data set all over again.** • **Very fast.**	• Can't deal with outcomes dependent on a combination of features. The assumption is that each is independent of each other-which can sometimes lower accuracy
K-Nearest Neighbours	• Online technique. Like a Naïve Bayes classifier, K-nearest Neighbours supports incremental training • Can handle complex numerical functions while still being easy to interpret. You can see exactly which neighbours are used for the final predictions. • Useful if data is difficult/expensive to collect. The scaling process can reveal which variables are unimportant for making predictions and thus can be	• Requires all training data to make predictions. This means K-nearest Neighbours can be very slow for large datasets and can require a lot of space • Finding correct scaling factors can be tedious and computationally expensive when there are millions of variables.

Support vector machines	• Very fast at classifying new data. There is no need to go through training data for new classifications. • Can work for a mixture of categorical and numerical data • SVMs are "Robust to high dimensionality", meaning they can work well even with a large number of features. • SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space. • Highly accurate.	• Black box technique. Unlike Bayesian Classifiers and Decision Trees, A SVM can be a highly effective classifier, but we may not be able to figure out *why* it makes those classifications. • Typically requires a very large dataset. Other methods, like decision trees, can still give interesting output for small data sets; this may not be the case with a SVM. • SVMs are not online. They will need to be updated every time new training data is to be incorporated.
Decision trees	• Very easy to interpret and explain. Decision Trees mirror human decision-making through their flowchart "if-then" style of splitting data down until the data is categorized. • Can be displayed graphically. A decision tree is a branching flowchart (just one that's been algorithmically tuned for optimal splits), and smaller decision trees displayed graphically can be easily interpreted even by non-technical people. • Can easily use both categorical and numerical data. In other classifiers, we will have to create a "dummy" variable to work around this. • Can deal with interactions of variables	• Decision Trees may not be as accurate as other classification algorithms as they have tendency to "overfit" the data. Nonetheless, there are various methods of "pruning" the tree to improve accuracy. • Decision Trees are not an online technique, meaning the whole tree must be recreated from scratch to incorporate new data, since the variables that optimally split the data could change. • With larger data sets the number of nodes on a decision tree can grow extremely large and complex, resulting in slow classifications.
Random Forest	• Good for parallel or distributed computing. • Usually has lower classification error and better f-scores than decision trees. • Almost always perform as well as or better than SVMs but are far easier for humans to understand. • Deal well with uneven data sets that have missing variables. • Gives good idea of which features in the data set are the most important for free.	• Decision Trees may not be as accurate as other classification algorithms as they have tendency to "overfit" the data. Nonetheless, there are various methods of "pruning" the tree to improve accuracy. • Decision Trees are not an online technique, meaning the whole tree must be recreated from scratch in order to incorporate new data, since the variables that optimally split the data could change. • With larger data sets the number of nodes on a decision tree can grow extremely large and complex, resulting in slow classifications.
Boosting	• It does not expect linear features or even features that interact linearly. • Because of way they are constructed it can handle very well high dimensional spaces as well as large number of training. • Allows significant tuning opportunities by change of parameters. Thus, making them significantly accurate than other models.	• Because of the various hyper parameters available to tune, it is somewhat difficult to set the right parameters unlike other algorithms. • May be prone to over fitting.
XGBoost	• XGBoosting supports user-defined objective functions with classification, regression and ranking problems. • Can use an objective function to measure the performance of a model • Can work with categorical and numerical data • Accuracy is better	• Because of the various hyper parameters available to tune, it is somewhat difficult to set the right parameters unlike other algorithms. • May be prone to overfitting

Bagging	• Method for improving results of classification algorithms. • Suitable means to increase efficiency of the classification algorithms, which have low values of precision and recall • Highly accurate	• It lacks illustrative ways of representation unlike other algorithms • Not simple and computationally complex.

8. Solution architectures

8.1. Functional Architecture



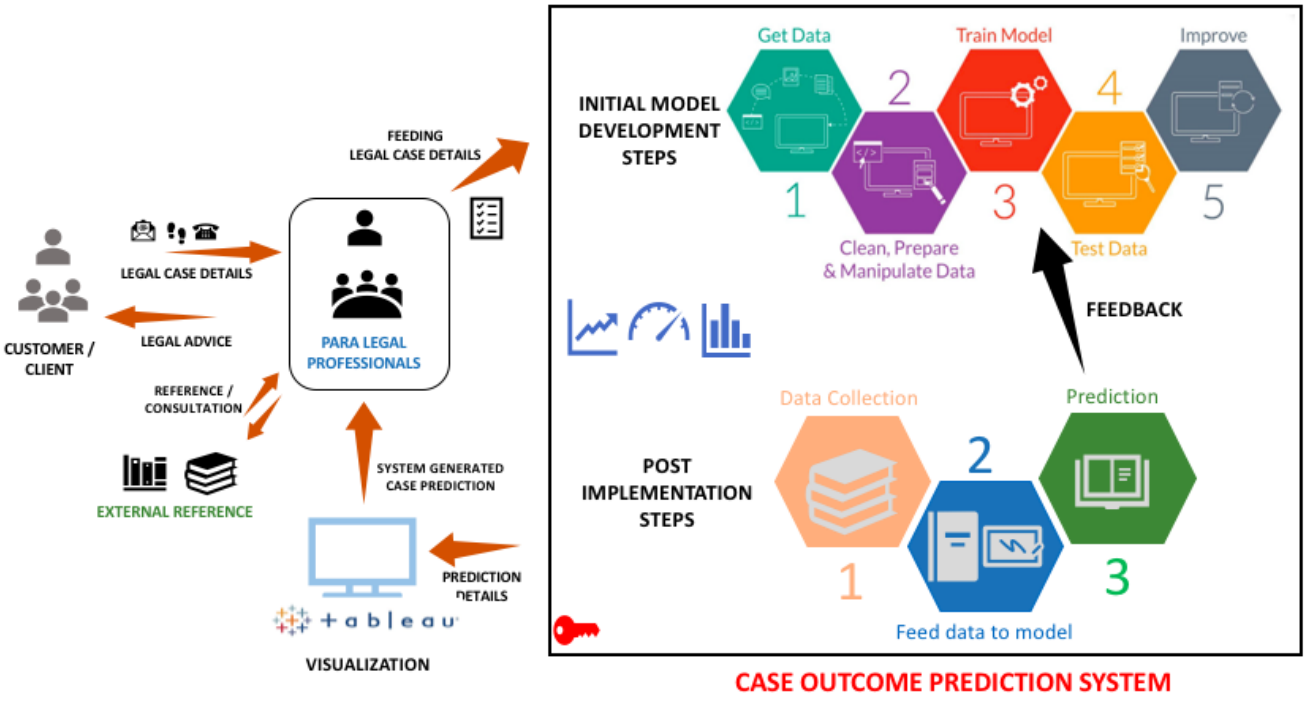
"Case Outcome Prediction System" as we are referring to our prediction model, will serve as a backend processing unit in the overall prediction of the outcome of legal cases.

Customers/Clients seeking legal suggestion will approach the Para legal professionals. Para legal professionals will enter the case details that will invoke the Case Outcome Prediction System. The results of the same will flow into a visualisation portal for better representation and interpretability. The legal professionals can read the outcome and ~~in~~ with reference to any external reference system/mechanism, the prediction outcome can enhance their decision making.

The outcome thus presented to the legal professionals will help them to decide if a legal case is viable in terms of their winning chances.

Our study focusses only on the functional unit of "Case Outcome Prediction System". The visualization module is an extension of the same and has been kept mainly from a reference perspective.

8.2. Technical architecture



We started with the initial process of data gathering where we collected features from the various judgement documents. The data had to be cleaned significantly and several features were eventually dropped to get 14 features that will help in predicting the target variable (Judgement).

The final dataset was split into training and testing. The initial model was built on the training data using multiple machine learning algorithms. The models were tested on the training data. The models were run through various performance checks to find the best model. Once the model is finalised the same was to be used for any future prediction requests.

9. Resampling / imbalance classification techniques

Class imbalance is one of the most prominent challenges in the machine learning area. In class imbalance problems, the number of examples of one class (minority class) is much smaller than the number of examples of the other classes, with the minority class being the class of greatest interest and that with the biggest error cost from the point of view of learning.

One of the approaches used to deal with class imbalance problems, called data approach, consists of resampling (subsampling or oversampling) the data to balance the classes before building the classifier.

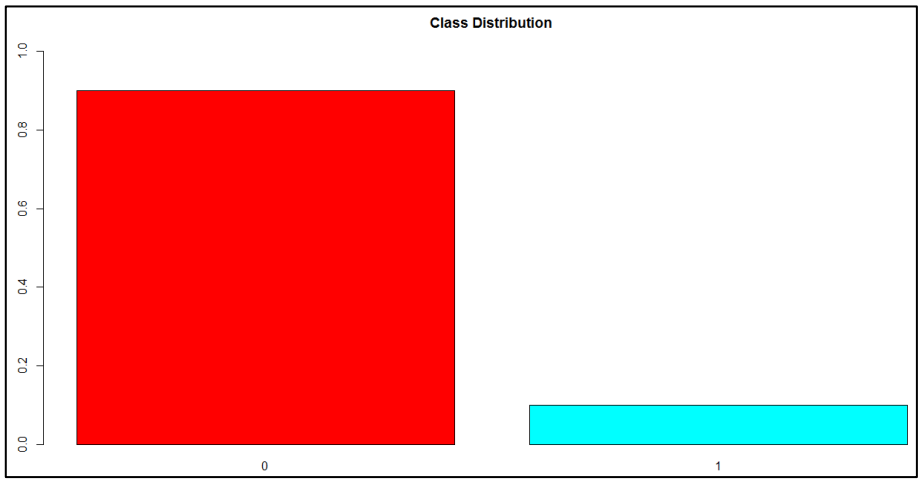
Resampling is the method that consists of drawing repeated samples from the original data samples. The method of Resampling is a nonparametric method of statistical inference which in other words refers that, the method of resampling does not involve the utilization of the generic distribution tables (e.g. normal distribution tables) to compute approximate p probability values.

Resampling involves the selection of randomized cases with replacement from the original data sample in such a manner that each number of the sample drawn has several cases that are like the original data sample. Due to

replacement, the drawn number of samples that are used by the method of resampling consists of repetitive cases.

Resampling generates a unique sampling distribution based on the actual data. The method of resampling uses experimental methods, rather than analytical methods, to generate the unique sampling distribution. The method of resampling yields unbiased estimates as it is based on the unbiased samples of all the possible results of the data studied by the researcher.

Our dataset as well had this issue where the minority class (Judgement = Guilty) was approximately 9% against the majority class (Judgement = Non-guilty).



We have tried the various machine learning models using the oversampling methods. The details are provided as below. Additionally, we have worked on a model based on Cost-Sensitive Training (Penalizing Algorithm). Resampling based on down-sampling (Under-sampling) has been attempted only for Random Forest.

10. Algorithm and implementation challenges

We handpicked few of the popular classification algorithms to develop the models on the original dataset. The details of the same are as included below.

1.1 Logistic Regression

Definition

It's a predictive modelling technique that allows us to identify factors affecting various categorical outcomes of interest (e.g., binary events like guilty or not-guilty) and the probability of those outcomes. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, guilty, etc.) or 0 (FALSE, failure, non-guilty, etc.).

The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a logit transformation of the probability of presence of the characteristic of interest.

Workout

We will use Logistic regression to build model and predict if the case announced in court will sentence the accused as guilty or not-guilty by considering features ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_gender, tot_sec, tot_accu, accu_gender and accu_age for the response variable - judgement that has two values, guilty and not-guilty.

The model returned the parameters as –

Log-Likelihood Ratio – Overall test of model significance based on chisq test below is highly significant indicating the likelihood of response depends upon the information provided. This implies that the null hypothesis of all Betas is zero is rejected and we conclude that at least one Beta is nonzero.

```
Model 1: judgement ~ ipc_420 + ipc_120b + ipc_471 + ipc_468 + ipc_34 +
    jud_gender + tot_sec + tot_accu + accu_gender + accu_age
Model 2: judgement ~ 1
  #Df  LogLik  Df  Chisq Pr(>Chisq)
1  11 -24.901
2   1 -45.617 -10 41.432  9.462e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pseudo R Square – Based on McFadden R Square, we conclude that 45.4% of the uncurtaining of the Intercept only model has been explained by the Full model. Thus, the goodness of fit is rather good.

llh	llhNull	G2	McFadden	r2ML	r2CU
-24.9007893	-45.6165819	41.4315853	0.4541286	0.2546050	0.5344257

Variable Importance – We can see from below that the most important features are ipc_471, ipc_468, jud_gender, tot_sec, tot_accu and accu_gender.

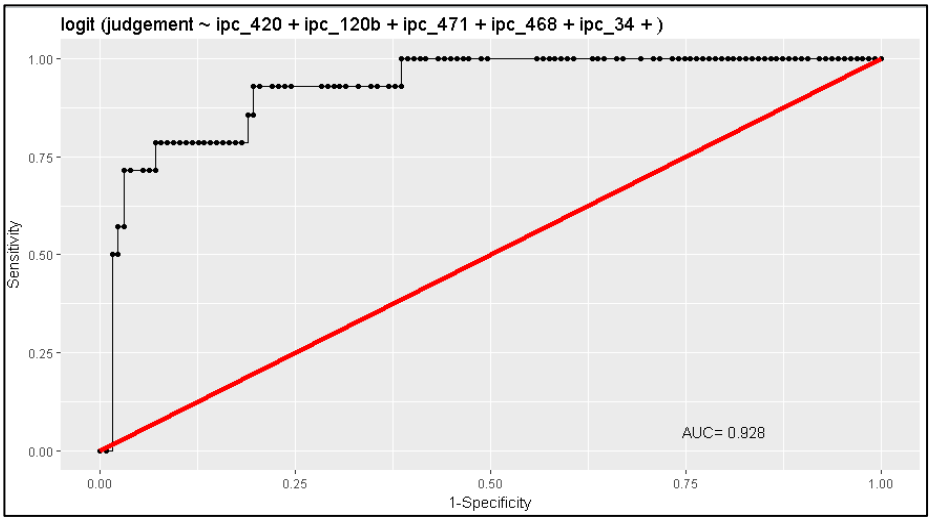
	Overall
ipc_4201	0.001264244
ipc_120b1	0.328154168
ipc_4711	1.875148682
ipc_4681	1.810797898
ipc_341	0.007793632
jud_gender1	1.024245431
tot_sec	1.652261795
tot_accu	1.515529311
accu_gender1	0.998149761
accu_age	0.060468610

Accuracy

For the training sample the accuracy came out to be 92.9% from the confusion matrix.

	Predicted	
Actual	0	1
0	124	3
1	7	7

ROC- PLOT



The ROC is a curve generated by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings while the AUC is the area under the ROC curve. As a rule of thumb, a model with good predictive ability should have an AUC closer to 1. Therefore, our model is having good predicting ability since AUC = .93.

Confusion Matrix on Validation Sample – The overall prediction accuracy thus obtained is ~88.52 %.

Confusion Matrix		
	Actual	
Predicted	**0**	**1**
0	50	2
1	5	4

Precision /Positive Predictive Value
The precision from the above prediction has been found to be ~96.15%.

Recall / Sensitivity
The recall from the above prediction is found to be ~90.91%.

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 50  2
         1  5  4

               Accuracy : 0.8852
                 95% CI : (0.7778, 0.9526)
    No Information Rate : 0.9016
    P-Value [Acc > NIR] : 0.7514

                  Kappa : 0.4709
 Mcnemar's Test P-Value : 0.4497

            Sensitivity : 0.9091
            Specificity : 0.6667
         Pos Pred Value : 0.9615
         Neg Pred Value : 0.4444
             Prevalence : 0.9016
         Detection Rate : 0.8197
   Detection Prevalence : 0.8525
      Balanced Accuracy : 0.7879

       'Positive' Class : 0
```

1.2 K-Nearest Neighbours

Definition

KNN is a non-parametric supervised learning technique in which we try to classify the data point to a given category with the help of training set. In simple words, it captures information of all training cases and classifies new cases based on a similarity.

Predictions are made for a new instance (x) by searching through the entire training set for the K most similar cases (neighbours) and summarizing the output variable for those K cases. In classification this is the mode (or most common) class value.

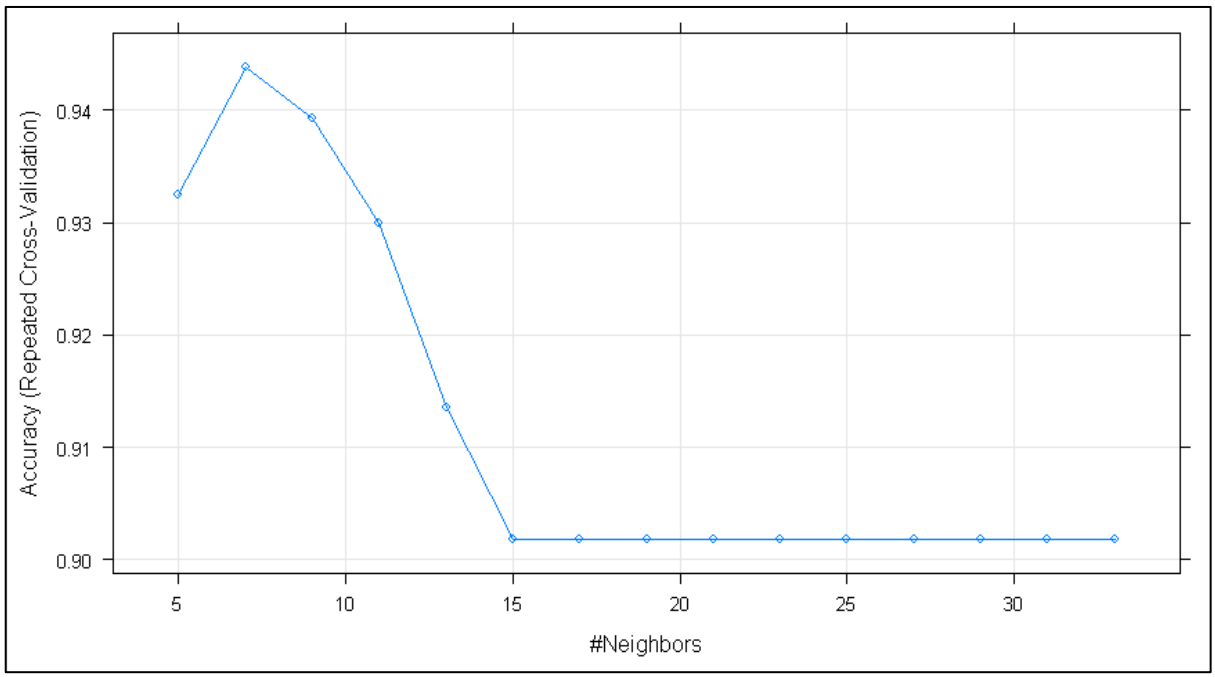
Workout

We fit the KNN model using the 14 variables (viz. ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_date, jud_gender, tot_sec, case_no, tot_accu , accu_gender, accu_no and accu_age) for the target variable judgement. We normalised and scaled dataset using 10-fold cross validation method that identified the best value of k = 7 to be used for model prediction.

The model returned the parameters as –

```
k-Nearest Neighbors

141 samples
 14 predictor
  2 classes: 'X0', 'X1'

Pre-processing: centered (124), scaled (124)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 127, 127, 127, 128, 127, 127, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9324420  0.4414103
   7  0.9438706  0.5363364
   9  0.9392674  0.4818648
  11  0.9298779  0.3924823
  13  0.9135287  0.1939980
  15  0.9017582  0.0000000
  17  0.9017582  0.0000000
  19  0.9017582  0.0000000
  21  0.9017582  0.0000000
  23  0.9017582  0.0000000
  25  0.9017582  0.0000000
  27  0.9017582  0.0000000
  29  0.9017582  0.0000000
  31  0.9017582  0.0000000
  33  0.9017582  0.0000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 7.
```

Its showing Accuracy and Kappa metrics result for different k value. From the results, it automatically selects best k-value. Here, our training model is choosing k = 7 as its final value.



Cross-Validation: fine tuning

Our model is trained using k=7, we will be predicting our validation sample using our training model as reference.

With this we are getting the model accuracy for test sample of 90.16%.

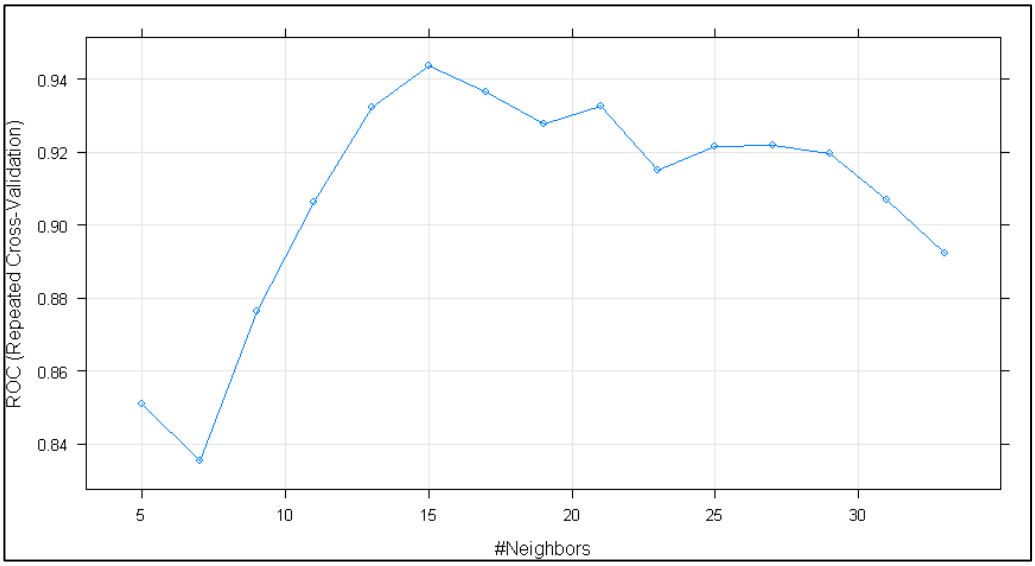
```
Confusion Matrix and Statistics

          Reference
Prediction X0 X1
        X0 53  4
        X1  2  2

               Accuracy : 0.9016
                 95% CI : (0.7981, 0.963)
    No Information Rate : 0.9016
    P-Value [Acc > NIR] : 0.6064

                  Kappa : 0.3488
 Mcnemar's Test P-Value : 0.6831

            Sensitivity : 0.9636
            Specificity : 0.3333
         Pos Pred Value : 0.9298
         Neg Pred Value : 0.5000
             Prevalence : 0.9016
         Detection Rate : 0.8689
   Detection Prevalence : 0.9344
      Balanced Accuracy : 0.6485

       'Positive' Class : X0
```

We now further tune our model using 2 more parameters twoclassSummary and classProbs set as True.

```
k-Nearest Neighbors

141 samples
 14 predictor
  2 classes: 'X0', 'X1'

Pre-processing: centered (124), scaled (124)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 126, 127, 126, 127, 127, 127, ...
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   5  0.8508547  0.9839744  0.3833333
   7  0.8352030  0.9918803  0.5000000
   9  0.8764423  0.9918803  0.4000000
  11  0.9060897  0.9918803  0.2833333
  13  0.9321047  0.9918803  0.2000000
  15  0.9437500  0.9972222  0.0000000
  17  0.9364850  1.0000000  0.0000000
  19  0.9276709  1.0000000  0.0000000
  21  0.9324786  1.0000000  0.0000000
  23  0.9150107  1.0000000  0.0000000
  25  0.9213675  1.0000000  0.0000000
  27  0.9217415  1.0000000  0.0000000
  29  0.9193910  1.0000000  0.0000000
  31  0.9068376  1.0000000  0.0000000
  33  0.8922543  1.0000000  0.0000000

ROC was used to select the optimal model using the largest value.
The final value used for the model was k = 15.
```

We can see from above that the optimal k value is now equal to 15.

Its showing Accuracy and Kappa metrics result for different k value. From the results, it automatically selects best k-value. Here, our training model is choosing k = 15 as its final value.



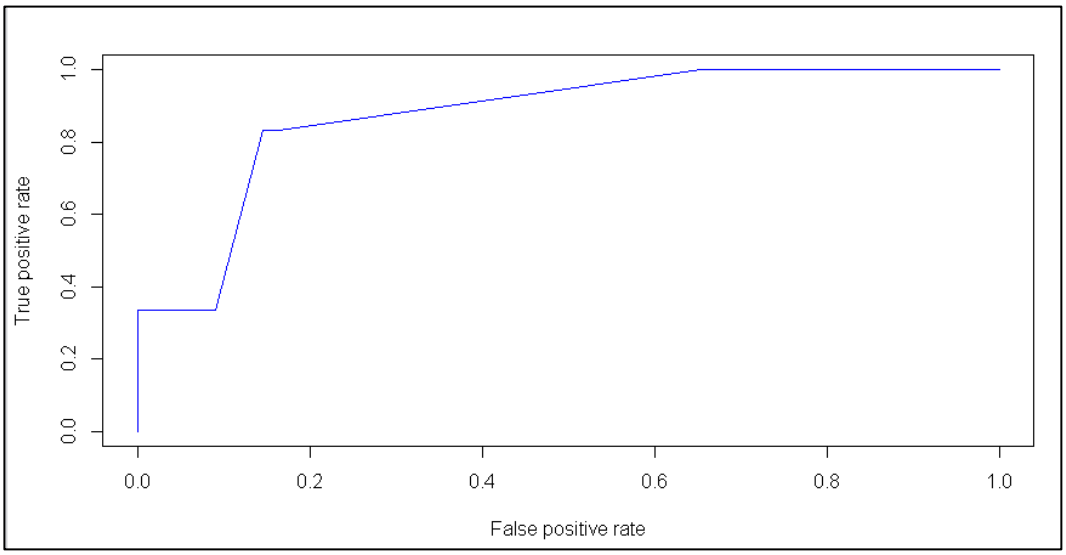
Cross-Validation: fine tuning

Accuracy

The algorithm using the 10-fold cross validation with normalising/scaling the dataset identified that the best values of k to be used is 15.

Using the values, the final model provided the below prediction.

ROC PLOT of the final tuned model:



Value of Area under the curve: 0.8727 shows that our model has good predicting capability.

The overall prediction accuracy thus obtained is ~93.44 %.

Confusion Matrix
Actual

Prediction	0	1
0	55	4
1	0	2

Precision /Positive Predictive Value
The precision from the above prediction has been found to be ~93.22%.

Recall / Sensitivity
The recall from the above prediction is found to be ~100%.

1.3 Support Vector Machines

Definition

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay on either side.

Workout

We fit the SVM model using the 14 variables (viz. ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_date, jud_gender, tot_sec, case_no, tot_accu , accu_gender, accu_no and accu_age) for the target variable judgement. The model returned the parameters as –

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.01724138

Number of Support Vectors:  41
```

There are 41 support vectors found. The Gamma value is 0.017 with a cost of 1.

Further predictions gave a count of 127 correct predictions with the accuracy of 90.02%.

This refers that our SVM model is accurately predicting 90.02% of the results.

Accuracy

We tried tuning our SVM model to find the best gamma and cost.

```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.1    1

- best performance: 0.05777222
```

The algorithm using the 10-fold cross validation identified that the best values of Gamma and Cost is 0.1 and 1 respectively.

Using the values, the final model provided the below prediction.

The overall prediction accuracy thus obtained is ~95.04 %.

Confusion Matrix		
	Actual	
Prediction	0	1
0	126	6
1	1	8

Precision /Positive Predictive Value
The precision from the above prediction has been found to be ~94.45%.

Recall / Sensitivity
The recall from the above prediction is found to be ~99.21%.

1.4 Naïve Bayes

Definition

A Naive Bayes classifier is an algorithm that uses Bayes' theorem to classify objects. Naive Bayes classifiers

assume strong, or naive, independence between attributes of data points. The key insight of Bayes' theorem is that the probability of an event can be adjusted as new data is introduced. These classifiers are widely used for machine learning because they are simple to implement. Naive Bayes is also known as simple Bayes or independence Bayes.

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Workout

We fit the Naive Bayes model using the 14 variables (viz. ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_date, jud_gender, tot_sec, case_no, tot_accu , accu_gender, accu_no and accu_age) for the target variable judgement.

Accuracy

The algorithm using the values, the final model provided the below prediction.

```
A priori probabilities:

        0         1
0.9202454 0.0797546
```

```
##########NB CLASSIFIER#################

CONFUSION MATRIX:

[[41 13]
 [ 2  5]]
             precision    recall  f1-score   support

          0       0.95      0.76      0.85        54
          1       0.28      0.71      0.40         7

avg / total       0.88      0.75      0.79        61

Accuracy of NB classifier model on test set: 75.41%
Precision: 27.78%
Recall: 71.43%
F1: 40.00%
```

The overall prediction accuracy thus obtained is ~75.4 %.

Confusion Matrix		
	Actual	
Prediction	**0**	**1**
0	41	13
1	2	5

Precision /Positive Predictive Value
The precision from the above prediction has been found to be ~75.93%.

Recall / Sensitivity
The recall from the above prediction is found to be ~95.35%.

1.5 Decision Trees

Definition

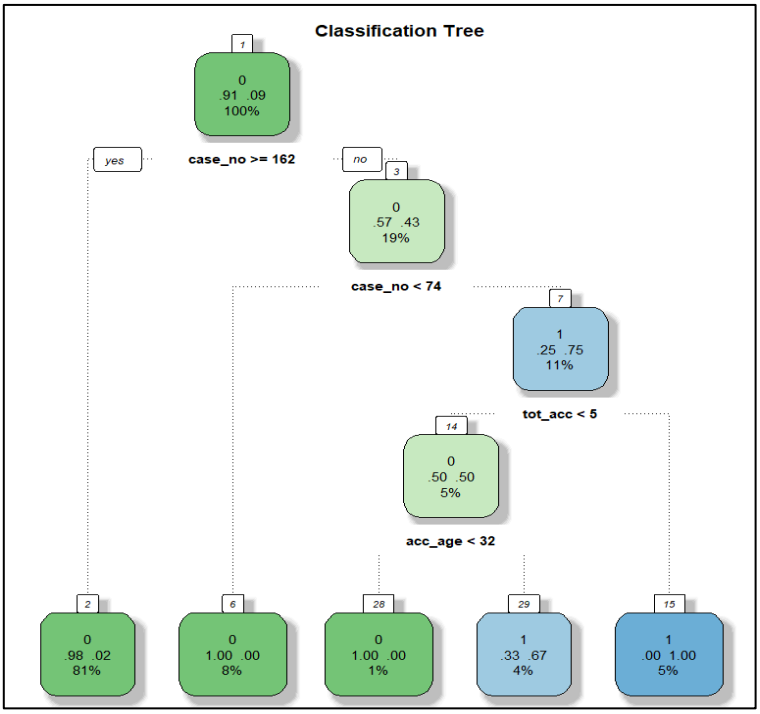
Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the dataset into two training and testing sample.

Workout

We fit the decision tree model using the 13 variables (viz. ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_gender, tot_sec, case_no, tot_accu , accu_gender, accu_no and accu_age) for the target variable judgement for the training dataset which has 150 observations. The model returned the parameters as –

```
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 150 14 0 (0.90666667 0.09333333)
   2) case_no>=161.5 122  2 0 (0.98360656 0.01639344) *
   3) case_no< 161.5 28 12 0 (0.57142857 0.42857143)
     6) case_no< 74.5 12  0 0 (1.00000000 0.00000000) *
     7) case_no>=74.5 16  4 1 (0.25000000 0.75000000)
      14) tot_acc< 5 8  4 0 (0.50000000 0.50000000)
        28) acc_age< 31.5 2  0 0 (1.00000000 0.00000000) *
        29) acc_age>=31.5 6  2 1 (0.33333333 0.66666667) *
      15) tot_acc>=5 8  0 1 (0.00000000 1.00000000) *
```

There were total of 29 nodes created by the decision tree model.

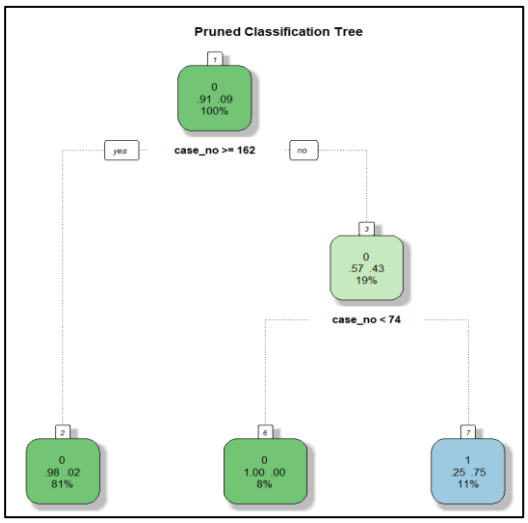


From the decision tree we can conclude that the accused convicted were dominant in node 7, 15 and 29.

```
Classification tree:
rpart(formula = Case_data_dev$judgement ~ ., data = Case_data_dev,
    method = "class", control = r.ctrl)

Variables actually used in tree construction:
[1] acc_age case_no tot_acc

Root node error: 14/150 = 0.093333

n= 150

        CP nsplit rel error  xerror    xstd
1 0.285714      0   1.00000 1.00000 0.25448
2 0.071429      2   0.42857 0.64286 0.20776
3 0.000000      4   0.28571 0.71429 0.21822
```

Here, the “rel error” is for the training set and “xerror” is for the cross-validation sample. Here when “nsplit” is 2 the “xerror” or classification error reduces and will consider the CP value of this split to prune the tree.

The output of the pruned classification tree is:



Confusion matrix (training sample)

```
          predict.class
judgement   0   1
        0 132   4
        1   2  12
```

Confusion matrix (testing sample)

```
          predict.class
judgement  0  1
        0 38  8
        1  1  5
```

The overall prediction accuracy achieved is 82.7%.

Confusion Matrix		
	Actual	
Prediction	**0**	**1**

0	38	8
1	1	5

Precision /Positive Predictive Value
The precision from the above prediction has been found to be ~82.61%.

Recall / Sensitivity
The recall from the above prediction is found to be ~97.45%.

1.6 Random Forest

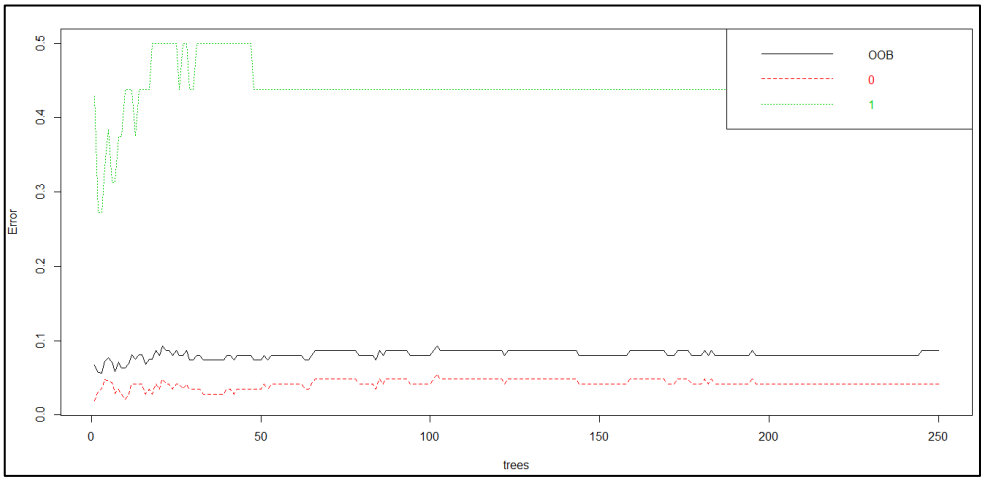
Definition

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

Workout

We fit the RF model using the 14 variables (viz. ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_date, jud_gender, tot_sec, case_no, tot_accu , accu_gender, accu_no and accu_age) for the target variable judgement. Below is random forest parameter selection

- ntree = 250 → as OOB hardly changes after 250 trees
- mtry = 3 → initially we took sqrt(total_no_of_features)
- nodesize = 3→ 1% of the total observation (202 observations)



Accuracy

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 33  2
         1  3  2

               Accuracy : 0.875
                 95% CI : (0.732, 0.9581)
    No Information Rate : 0.9
    P-Value [Acc > NIR] : 0.7937

                  Kappa : 0.375
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.5000
            Specificity : 0.9167
         Pos Pred Value : 0.4000
         Neg Pred Value : 0.9429
             Prevalence : 0.1000
         Detection Rate : 0.0500
   Detection Prevalence : 0.1250
      Balanced Accuracy : 0.7083

       'Positive' Class : 1
```

Cross Validation with Parameter Tuning

We used 10-fold Cross Validation with mtry=2, 3 and 4 and we got best mtry=2 with high accuracy

```
Random Forest

135 samples
 13 predictor
  2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 122, 122, 122, 121, 122, 122, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.9405861  0.3894297
  3     0.9263004  0.3448953
  4     0.9263004  0.3448953

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Accuracy

```
Confusion Matrix and Statistics

          Reference
Prediction X0 X1
        X0 58  4
        X1  0  5

               Accuracy : 0.9403
                 95% CI : (0.8541, 0.9835)
    No Information Rate : 0.8657
    P-Value [Acc > NIR] : 0.04368

                  Kappa : 0.684
 Mcnemar's Test P-Value : 0.13361

            Sensitivity : 0.55556
            Specificity : 1.00000
         Pos Pred Value : 1.00000
         Neg Pred Value : 0.93548
             Prevalence : 0.13433
         Detection Rate : 0.07463
   Detection Prevalence : 0.07463
      Balanced Accuracy : 0.77778

       'Positive' Class : X1
```

Confusion Matrix

	Actual	
Prediction	**0**	**1**

0	58	4
1	0	5

Precision /Positive Predictive Value
The precision from the above prediction has been found to be ~93.55%.

Recall / Sensitivity
The recall from the above prediction is found to be ~100%.

1.7 Bagging

Bootstrap aggregating, also called bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting

Workout

We fit the RF and CART model by using Bagging Technique using 14 variables (viz. ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_date, jud_gender, tot sec, case_no, tot_accu , accu_gender, accu_no and accu_age) for the target variable judgement. Below are Bagging results

Bagging on Training Set

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 142   6
         1   4  10

               Accuracy : 0.9383
                 95% CI : (0.8894, 0.97)
    No Information Rate : 0.9012
    P-Value [Acc > NIR] : 0.06716

                  Kappa : 0.6328
 Mcnemar's Test P-Value : 0.75183

            Sensitivity : 0.62500
            Specificity : 0.97260
         Pos Pred Value : 0.71429
         Neg Pred Value : 0.95946
             Prevalence : 0.09877
         Detection Rate : 0.06173
   Detection Prevalence : 0.08642
      Balanced Accuracy : 0.79880

       'Positive' Class : 1
```

Bagging on Testing Set

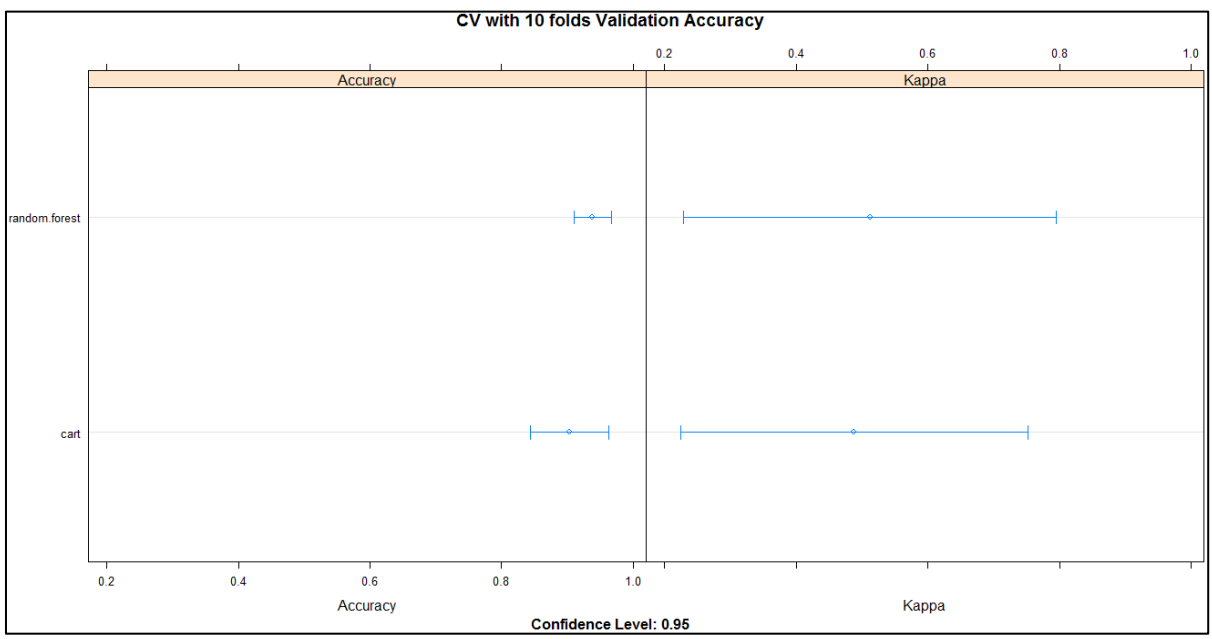
```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 35  1
         1  1  3

               Accuracy : 0.95
                 95% CI : (0.8308, 0.9939)
    No Information Rate : 0.9
    P-Value [Acc > NIR] : 0.2228

                  Kappa : 0.7222
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.7500
            Specificity : 0.9722
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.9722
             Prevalence : 0.1000
         Detection Rate : 0.0750
   Detection Prevalence : 0.1000
      Balanced Accuracy : 0.8611

       'Positive' Class : 1
```

K-Fold Cross Validation with 10 Folds

When we did Cross Validation with 10-Folds on RF and CART, we got 93.85% of accuracy

```
Call:
summary.resamples(object = bagging_results)

Models: cart, random.forest
Number of resamples: 10

Accuracy
                  Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
cart          0.7647059 0.8768382 0.9333333 0.9032108 0.9402574    1    0
random.forest 0.8750000 0.9343750 0.9375000 0.9385539 0.9411765    1    0

Kappa
                   Min.    1st Qu.    Median      Mean   3rd Qu. Max. NA's
cart          -0.06666667 0.2551910 0.5337398 0.4877634 0.6378143    1    0
random.forest -0.06666667 0.1585366 0.6363636 0.5116803 0.6382979    1    0
```

K-Fold Accuracy Plot



Accuracy of the model is

Confusion Matrix		
	Actual	
Prediction	0	1
0	35	1
1	1	3

Precision /Positive Predictive Value
The precision from the above prediction has been found to be ~97.22%.

Recall / Sensitivity
The recall from the above prediction is found to be ~97.22%.

1.8 XG Boosting

Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms. So, what makes it fast is its capacity to do parallel computation on a single machine.

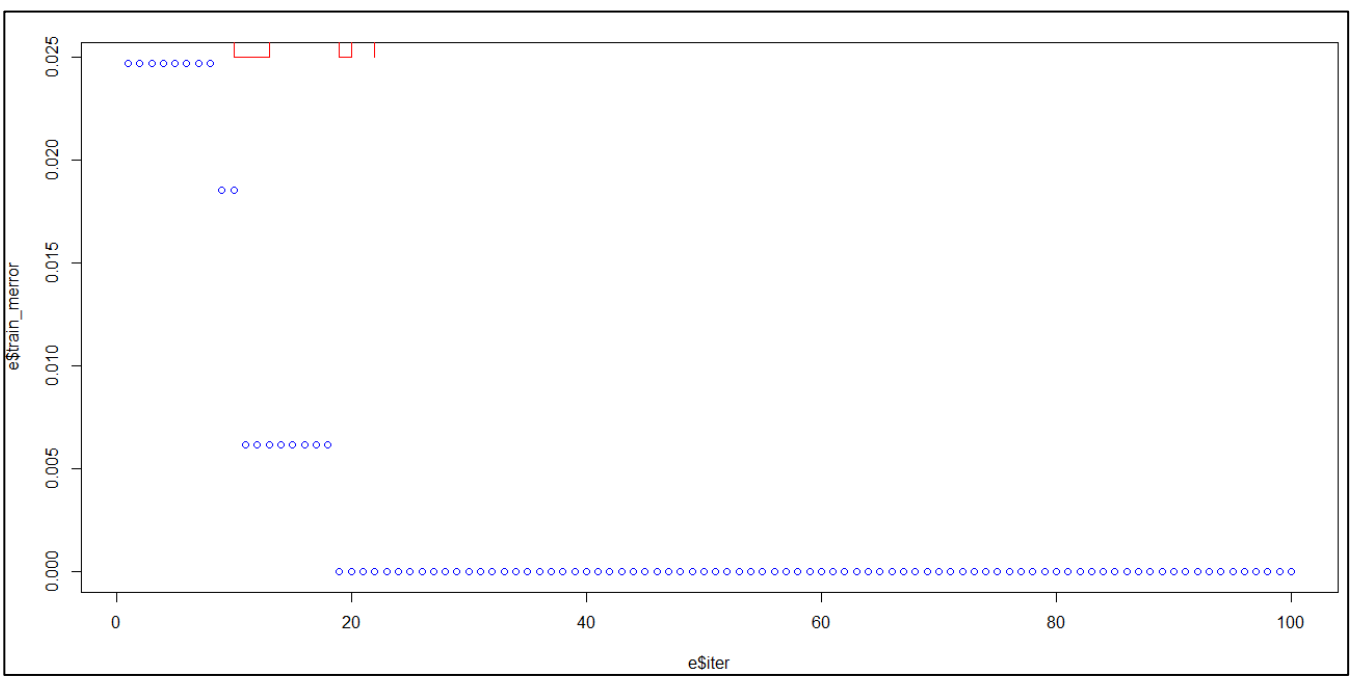
Workout

We fit the eXtreme Gradient Boosting model by using 14 variables (viz. ipc_420, ipc_120b, ipc_471, ipc_468, ipc_34, jud_date, jud_gender, tot_sec, case_no, tot_accu, accu_gender, accu_no and accu_age) for the target variable judgement. Below are XgGBM results.

We first ran with standard parameters for xGBoost and below is the output of the model

```
##### xgb.Booster
Handle is invalid! Suggest using xgb.Booster.complete
raw: 89.6 Kb
call:
  xgb.train(params = xgb_params, data = train_matrix, nrounds = 100,
    watchlist = watchlist, eta = 0.3, seed = 333)
params (as set within xgb.train):
  objective = "multi:softprob", eval_matric = "mlgloss", num_class = "2", eta = "0.3", seed = "333", silent = "1"
callbacks:
  cb.print.evaluation(period = print_every_n)
  cb.evaluation.log()
# of features: 13
niter: 100
nfeatures : 13
evaluation_log:
    iter train_merror test_merror
       1     0.024691       0.125
       2     0.024691       0.125
---
      99     0.000000       0.050
     100     0.000000       0.050
```

When we plot the log-loss of test error, we found that the model is overfitting and test errors are drastically increasing on the above selected parameters.



Model Accuracy before parameter tuning

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 35  1
         1  1  3

               Accuracy : 0.95
                 95% CI : (0.8308, 0.9939)
    No Information Rate : 0.9
    P-Value [Acc > NIR] : 0.2228

                  Kappa : 0.7222
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.7500
            Specificity : 0.9722
         Pos Pred Value : 0.7500
         Neg Pred Value : 0.9722
             Prevalence : 0.1000
         Detection Rate : 0.0750
   Detection Prevalence : 0.1000
      Balanced Accuracy : 0.8611

       'Positive' Class : 1
```

Conclusion

Accuracy of the model in testing data is 92.10%. The model seems overfitting and the accuracy cannot be replayed on the test data. Thus, fine tuning of the parameter is essential.

Confusion Matrix		
	Actual	
Prediction	**0**	**1**
0	35	1
1	1	3

Precision /Positive Predictive Value

The precision from the above prediction has been found to be ~97.22%.

Recall / Sensitivity

The recall from the above prediction is found to be ~97.22%.

11. Hyperparameter tuning

In any machine learning model, there are choices that one can make to define the model structure for the model to perform optimally. In most times, this becomes a hit and trial method, during which we come across the best parameters that best delivers the performance of the model. These parameters are identified by the exploration done by the machine. The parameters which define the architecture of the method are referred as hyperparameters and this process of searching for the ideal model architecture is referred as hyperparameter tuning.

Some of design questions that can be helped by the hyper parameters are viz.

- The maximum depth allowed for the decision tree.
- Degree of polynomial features to be used linear model
- Minimum number of samples required in leaf node in the decision tree.
- How many trees to be included in the random forest
- How many neurons to have neural network layer
- How many layers to have in neural network
- Learning rate to set for gradient descent

Hyperparameters are not the model parameters and cannot be directly trained from the data. Model parameters are learned during training, for instance while trying to optimize a loss function using gradient descent. The process for learning parameter values is shown generally below.

Model-based learning

Use the input data

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \dots & x_{1,n} \\ x_{2,0} & x_{2,1} & \dots & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,0} & x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \text{ and } \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$$

To learn a set of parameters

$$\begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix}$$

Which yield a **generalized** function

$$f(x;\theta) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Capable of predicting values or classes on new input data

$$f(x_i;\theta) = 39$$

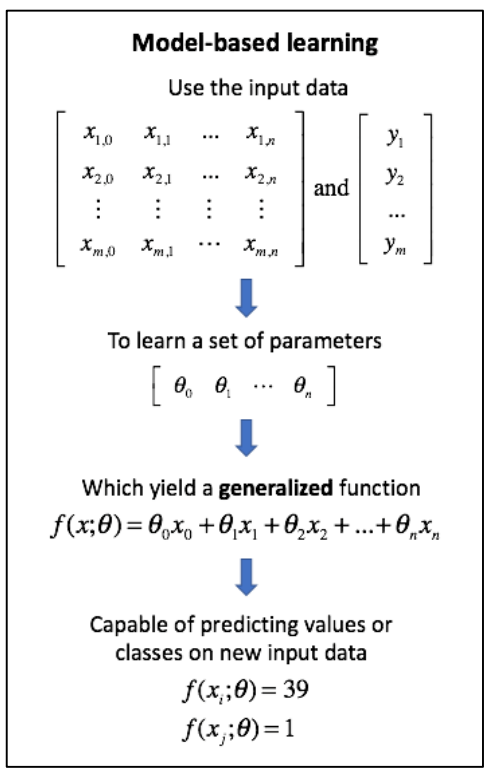
$$f(x_j;\theta) = 1$$

In any machine learning model, there are choices that one can make to define the model structure for the model to perform optimally. In most times, this becomes a hit and trial method, during the course of which we come across the best parameters that best delivers the performance of the model. These parameters are identified by the exploration done by the machine. The parameters which define the architecture of the method are referred as hyperparameters and this process of searching for the ideal model architecture is referred as hyperparameter tuning.

Some of design questions that can be helped by the hyper parameters are viz.

- The maximum depth allowed for the decision tree.
- Degree of polynomial features to be used linear model
- Minimum number of samples required in a leaf node in the decision tree.
- How many trees to be included in the random forest
- How many neurons must be there in a neural network layer
- How many layers to have in neural network
- Learning rate to be set for gradient descent

Hyperparameters are not the model parameters and cannot be directly trained from the data. Model parameters are learned during training, for instance while trying to optimize a loss function using gradient descent. The process for learning parameter values is shown generally below.



While the model parameters define how to transform the input data into the desired output, the hyper parameters define how the model is structured. Developments have still not been that impactful to the ways to update the hyperparameter to reduce the loss, the general approach is still to experiment and figure out what works best.

The process usually followed:

- Defining a model
- Defining the range of possible values for all hyperparameters
- Defining a method for sampling hyperparameter values
- Defining an evaluative criterion to judge the model
- Defining a cross-validation method

To highlight the usage of **Hyperparameter Tuning** we have tried the same with the Support Vector Machine and XGBoost models. Details about the same are as follows:

Support Vector Machine

```
##################################################
# HYPERPARAMETER TUNING ON SVM USING GRIDSEARCH #
##################################################

Name: judgement, Length: 202, dtype: category
Categories (2, int64): [0, 1]
# Tuning hyper-parameters for precision
()
Best parameters set found on development set:
()
{'kernel': 'rbf', 'C': 1, 'gamma': 0.001}
()
Grid scores on development set:
()
0.894 (+/-0.197) for {'kernel': 'rbf', 'C': 1, 'gamma': 0.001}
0.454 (+/-0.014) for {'kernel': 'rbf', 'C': 1, 'gamma': 0.0001}
0.816 (+/-0.258) for {'kernel': 'rbf', 'C': 10, 'gamma': 0.001}
0.708 (+/-0.338) for {'kernel': 'rbf', 'C': 10, 'gamma': 0.0001}
0.755 (+/-0.227) for {'kernel': 'rbf', 'C': 100, 'gamma': 0.001}
0.762 (+/-0.217) for {'kernel': 'rbf', 'C': 100, 'gamma': 0.0001}
0.755 (+/-0.227) for {'kernel': 'rbf', 'C': 1000, 'gamma': 0.001}
0.700 (+/-0.069) for {'kernel': 'rbf', 'C': 1000, 'gamma': 0.0001}
0.877 (+/-0.277) for {'kernel': 'linear', 'C': 1}
0.779 (+/-0.331) for {'kernel': 'linear', 'C': 10}
0.715 (+/-0.275) for {'kernel': 'linear', 'C': 100}
0.719 (+/-0.288) for {'kernel': 'linear', 'C': 1000}
()
Detailed classification report:
()
The model is trained on the full development set.
The scores are computed on the full evaluation set.
()
             precision    recall  f1-score   support

          0       0.95      0.98      0.96        54
          1       0.80      0.57      0.67         7

avg / total       0.93      0.93      0.93        61
```

Using Grid Search we tried to tune our SVM model. The classification report as obtained shows the total scores as follows:

- Precision – 93%
- Recall – 93%
- F1-score – 0.93

```
# Tuning hyper-parameters for recall
()
Best parameters set found on development set:
()
{'kernel': 'linear', 'C': 1}
()
Grid scores on development set:
()
0.769 (+/-0.228) for {'kernel': 'rbf', 'C': 1, 'gamma': 0.001}
0.500 (+/-0.000) for {'kernel': 'rbf', 'C': 1, 'gamma': 0.0001}
0.757 (+/-0.220) for {'kernel': 'rbf', 'C': 10, 'gamma': 0.001}
0.723 (+/-0.302) for {'kernel': 'rbf', 'C': 10, 'gamma': 0.0001}
0.709 (+/-0.117) for {'kernel': 'rbf', 'C': 100, 'gamma': 0.001}
0.754 (+/-0.226) for {'kernel': 'rbf', 'C': 100, 'gamma': 0.0001}
0.709 (+/-0.117) for {'kernel': 'rbf', 'C': 1000, 'gamma': 0.001}
0.746 (+/-0.230) for {'kernel': 'rbf', 'C': 1000, 'gamma': 0.0001}
0.783 (+/-0.328) for {'kernel': 'linear', 'C': 1}
0.719 (+/-0.253) for {'kernel': 'linear', 'C': 10}
0.711 (+/-0.252) for {'kernel': 'linear', 'C': 100}
0.746 (+/-0.263) for {'kernel': 'linear', 'C': 1000}
()
Detailed classification report:
()
The model is trained on the full development set.
The scores are computed on the full evaluation set.
()
             precision    recall  f1-score   support

          0       0.95      0.96      0.95        54
          1       0.67      0.57      0.62         7

avg / total       0.91      0.92      0.92        61

Accuracy of SVM classifier on test set: 91.80%
Precision: 66.67%
Recall: 57.14%
```

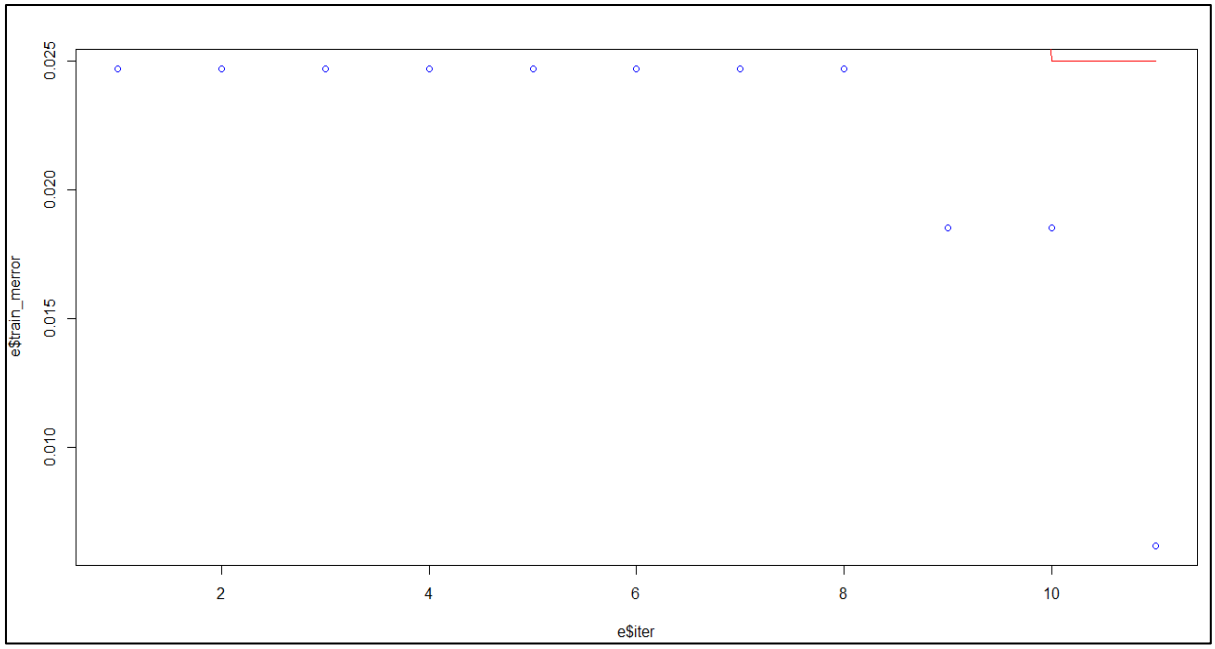
The hyperparameter for the best Recall and the final total recall was found to be 93% with 57.14% in favour of Judgement value of 1(Guilty).

XGBoost

We fine-tuned the parameters and re-built the xGBoost model and below is the result of models with hyper parameter tuning.

```
##### xgb.Booster
Handle is invalid! Suggest using xgb.Booster.complete
raw: 14.9 Kb
call:
  xgb.train(params = xgb_params, data = train_matrix, nrounds = 11,
    watchlist = watchlist, eta = 0.3, gamma = 0.002, max.depth = 6,
    subsample = 1, colsample_bytree = 1, missing = NA, seed = 333)
params (as set within xgb.train):
  objective = "multi:softprob", eval_matric = "mlgloss", num_class = "2", eta = "0.3", gamma = "0.002", max_depth = "6",
 subsample = "1", colsample_bytree = "1", missing = "NA", seed = "333", silent = "1"
callbacks:
  cb.print.evaluation(period = print_every_n)
  cb.evaluation.log()
# of features: 13
niter: 11
nfeatures : 13
evaluation_log:
    iter train_merror test_merror
       1     0.024691       0.125
       2     0.024691       0.125
---
      10     0.018519       0.025
      11     0.006173       0.025
```

When we plot the test error log loss as shown below, we found that the model is now not over fitted and minimal loss of error 0.025.



Accuracy and Confusion Matrix on Test Data

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 35  0
         1  1  4

               Accuracy : 0.975
                 95% CI : (0.8684, 0.9994)
    No Information Rate : 0.9
    P-Value [Acc > NIR] : 0.08047

                  Kappa : 0.875
 Mcnemar's Test P-Value : 1.00000

            Sensitivity : 1.0000
            Specificity : 0.9722
         Pos Pred Value : 0.8000
         Neg Pred Value : 1.0000
             Prevalence : 0.1000
         Detection Rate : 0.1000
   Detection Prevalence : 0.1250
      Balanced Accuracy : 0.9861

       'Positive' Class : 1
```

After hyperparameter tuning, we noticed that accuracy has reached 97.5%.

12. Proposed Algorithms and their advantages

Business Challenge/ Problem	Algorithms/ Analytics	Value addition
Data preparation	Statistical	Set of algorithms to prepare uncorrelated and denoised data
Outcome prediction		
Outcome validation		
Natural language processing	NLTK	Set of natural language processing tools to find patterns in associated features in transaction data
Technology	R and Python	

13. Features importance

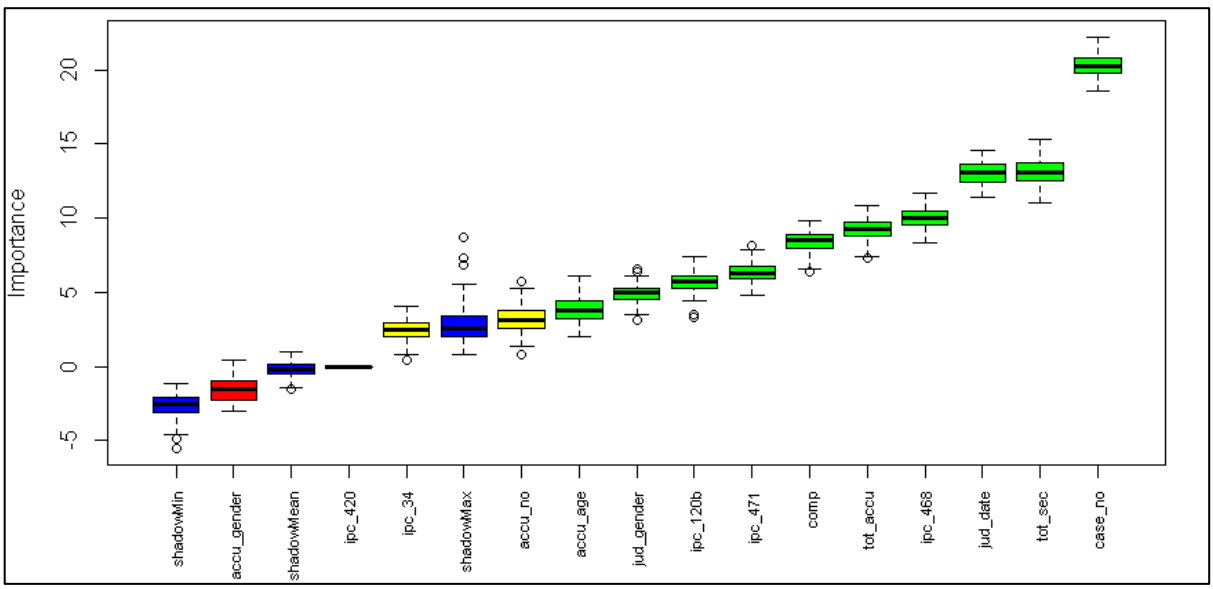
We selected 14 features to predict target variable i.e., judgement using legal domain knowledge of our domain expert.

We are just trying to analyse the contribution of each feature selected towards model building and prediction of response variable.

For feature importance, we are using Boruta algorithm which is a wrapper built around the random forest classification algorithm. It tries to capture all the important, interesting features out of the 14 features selected from domain knowledge in our dataset with respect to an outcome variable.

```
Boruta performed 99 iterations in 17.06836 secs.
 10 attributes confirmed important: accu_age, case_no, comp, ipc_120b, ipc_468 and 5
more;
 2 attributes confirmed unimportant: accu_gender, ipc_420;
 2 tentative attributes left: accu_no, ipc_34;
```

Boruta gives a crystal-clear call on the significance of variables in a dataset. In this case, out of 14 attributes, 2 of them are considered unimportant and 10 are confirmed. 2 attributes are designated as tentative. Tentative attributes have importance so close to their best shadow attr ibutes that Boruta is not able to decide with the desired confidence in default number of rand om forest runs.



Blue boxplots correspond to minimal, average and maximum Z score of a shadow attribute. Red, yellow and green boxplots represent Z scores of rejected, tentative and confirmed attribu tes respectively.

The tentative attributes will be classified as confirmed or rejected by comparing the median Z score of the attributes with the median Z score of the best shadow attribute.

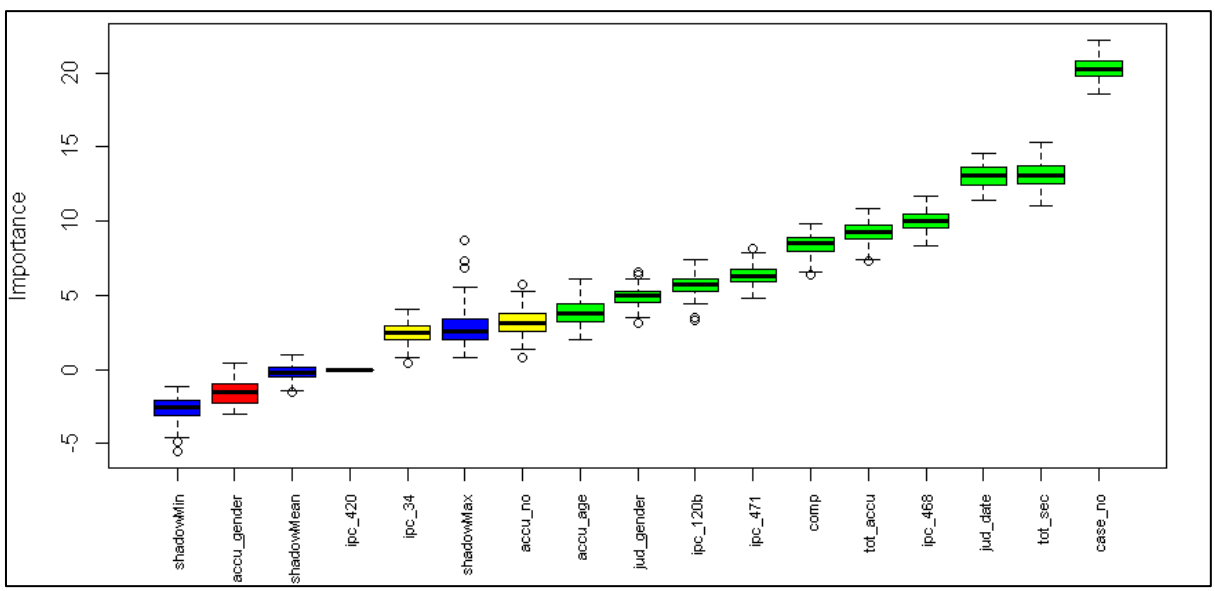
We selected 14 features to predict target variable i.e., judgement using legal domain knowledge of our domain expert.

We are trying to analyse the contribution of each feature selected towards model building and prediction of response variable.

For feature importance, we are using Boruta algorithm which is a wrapper built around the random forest classification algorithm. It tries to capture all the important, interesting features out of the 14 features selected from domain knowledge in our dataset with respect to an outcome variable.

```
Boruta performed 99 iterations in 17.06836 secs.
 10 attributes confirmed important: accu_age, case_no, comp, ipc_120b, ipc_468 and 5
more;
 2 attributes confirmed unimportant: accu_gender, ipc_420;
 2 tentative attributes left: accu_no, ipc_34;
```

Boruta gives a crystal-clear call on the significance of variables in a dataset. In this case, ou t of 14 attributes, 2 of them are considered unimportant and 10 are confirmed. 2 attributes ar e designated as tentative. Tentative attributes have importance so close to their best shadow attributes that Boruta is not able to decide with the desired confidence in default number of random forests runs.



Blue boxplots correspond to minimal, average and maximum Z score of a shadow attribute. Red, yellow and green boxplots represent Z scores of rejected, tentative and confirmed attri butes respectively.

The tentative attributes will be classified as confirmed or rejected by comparing the median Z score of the attributes with the median Z score of the best shadow attribute.

Boruta result plot after the classification of tentative attributes

```
Boruta performed 99 iterations in 17.06836 secs.
Tentatives roughfixed over the last 99 iterations.
 11 attributes confirmed important: accu_age, accu_no, case_no, comp, ipc_120b and 6
more;
 3 attributes confirmed unimportant: accu_gender, ipc_34, ipc_420;
```

List of confirmed attributes derived are

```
[1] "ipc_120b"   "ipc_471"    "ipc_468"    "jud_gender" "jud_date"   "tot_sec"    "case_no"
[8] "comp"       "tot_accu"   "accu_no"    "accu_age"
```

Final data frame of the feature importance derived from Boruta.

	meanImp	medianImp	minImp	maxImp	normHits	decision
ipc_420	0.000000	0.000000	0.0000000	0.0000000	0.0000000	Rejected
ipc_120b	5.696416	5.745901	3.2875414	7.3820280	0.9393939	Confirmed
ipc_471	6.354246	6.314182	4.7968047	8.1464092	0.9696970	Confirmed
ipc_468	10.026249	9.991072	8.3699037	11.6477026	1.0000000	Confirmed
ipc_34	2.486586	2.510379	0.4285620	4.0908279	0.4343434	Rejected
jud_gender	4.928849	4.952507	3.0968981	6.6116472	0.8888889	Confirmed
jud_date	13.065262	13.092875	11.4080454	14.5366091	1.0000000	Confirmed
tot_sec	13.164226	13.114093	11.0178478	15.3660702	1.0000000	Confirmed
case_no	20.255807	20.231048	18.5720286	22.1937364	1.0000000	Confirmed
comp	8.455193	8.561986	6.4237923	9.8310976	0.9898990	Confirmed
tot_accu	9.197894	9.239623	7.3563821	10.8350910	1.0000000	Confirmed
accu_gender	-1.585160	-1.498666	-3.0270235	0.4286216	0.0000000	Rejected
accu_no	3.150059	3.115759	0.8099369	5.7731835	0.5959596	Confirmed
accu_age	3.860509	3.830035	2.0369391	6.0698373	0.7777778	Confirmed

Thus, we see that using this algorithm ipc_420, ipc_34 and accused gender seems to be less important features as compared to others in the feature list.

14. Comparative Results

Results across models run on original data

Model	Accuracy (%)	Precision (%)	Recall (%)
Bagging	92%	97%	97%
Decision Trees (Gini)	82%	82%	97%
Gradient Boosting	**95%**	**100%**	**57%**
K-Nearest Neighbour	93%	93%	100%
Logistic Regression	88%	96%	91%
Naïve Bayes	75%	76%	95%
Random Forest	94%	93%	100%
Support Vector Machines	**95%**	**94%**	**99%**

Results across models run on resampled data

Model	Accuracy (%)	Precision (%)	Recall (%)
Bagging	95%	83%	71%
Decision Trees (Gini)	95%	100%	57%
Gradient Boosting	**95%**	**100%**	**57%**
K-Nearest Neighbour	93%	80%	57%
Logistic Regression	93%	80%	57%
Naïve Bayes	75%	28%	71%
Random Forest	100%	100%	100%
Support Vector Machines	**95%**	**100%**	**57%**

15. Business Value

Looking at the various AI methods and the value it can generate; we are comfortable to record that a field as abstract and uncertain as law can get some significant assistance from AI techniques and technologies. The legal system in general or perceivably imprints an image of long arduous process of constant pursuance. Be it the processes involved or the basic reason of multiple cross checking to avoid a wrong assessment; legal process requires significant effort taking from attorneys as well as the parties involved.

The “Predicting outcome of Legal Cases” should help in the early assessments of any case. This will help in assessing whether the case is viable to be fought in court or alternative settlement is a better option. The model developed as part of this study will help the Para legal representatives in assessing their cases to see the winning proportion of the case and can advise their customers accordingly. This work is not **developed to**

override the cognitive thinking of the trained attorneys but should augment in their overall decision-making process.

Additionally, legal domain being such an uncertain field is in a very nascent state with technology adaptation and given the ease of outcome prediction that can be delivered by our model; attorneys or law firms will be better equipped in decision making from a case winning perspective. This also saves significant amount of preparation and paper work which otherwise will be required if the cases are directly filed in courts. Our model will help these situations by providing a fair approximate assessment on the case prediction that can provide inputs on the future course of the case's justifiability.

References

Henri Stern, Michael Zhu, 2018. Predicting Supreme Court decisions using supervised learning, *Preprint*.

Caryn D., Teppo F., Stuart K., Roger K., 2018. The law and big data, *Preprint*.

Deep-Burn Project: Annual Report for 2009, Idaho National Laboratory, Sept. 2009.

Mathew Hutson, 2018. "Artificial intelligence prevails at predicting Supreme Court decisions" http://www.sciencemag.org/news/2017/05/artificial-intelligence-prevails-predicting-supreme-court-decisions.

Harry Surden, 2018. "Predicting Supreme Court decisions using artificial intelligence" http://www.harrysurden.com/wordpress/archives/248

Harry Surden, 2014. "Machine learning and law", Washington Law Design, Vol 89, p. 29.

Harry Surden, 2017. "Values embedded in legal artificial intelligence", University of Corolado Law Legal Studies Research Paper, No.17, p. 6.