Stock Market Analysis and Time Series Predictions

Vignesh Kumar S
Department of Chemical Engineering
Indian Institute of Technology, Madras
Email address: ch18b118@smail.iitm.ac.in

***Abstract*—The purpose of this article is to analyse a new type of dataset, called the Time Series where the order of data plays a key role using techniques covered through the coursework and it's extension. In this work, we demonstrate the application of the learnings illustrated in the world of finance through Stock Prices of different companies. Due to the complex nature and the abstractness involved in Stock Market, it is often necessary to forego a detailed analysis before investments to make informed choices. Using the concepts learned in the course, we try obtaining insights and analysing different stocks in a large amount of data comprehensively. We then try to forecast our predictions that could help in analysing risks and benefits of an investment.**

I. Introduction

In this paper, we will study Time Series Analysis, a technique used to analyze data where sequence is of importance, usually data involving date and time. We will use this technique to forecast a stock of interest that was analysed by weighing risks and returns. This analysis would aid in obtaining insights among a large dataset and hopefully uncover the trend behind a stock of interest.

The stock market allows numerous buyers and sellers of securities to meet, interact, and transact. Stock analysis is important for any investors and traders to make buying and selling decisions. By studying and evaluating past and current data, one could attempt to gain an edge in the markets by making informed decisions.

The dataset used for this problem comprises opening, high, low and closing prices of six stocks in India. We try to analyse each of those stocks to compare and contrast each of them. We then attempt to forecast the predictions of a stock that could suit a particular candidate using time series analysis and the techniques learnt in the course.

This work represents implementing the concepts learnt in the course and analysis of Time Series data in the finance world. This is purely written in academic interest and should not be used for personal decisions without professional advice.

II. Time Series Analysis

A time series is a sequence of data points that occur in successive order over some period of time. Since Time series data have a natural temporal ordering, this can be contrasted with cross-sectional data which captures a point-in-time and has no natural ordering of the observations. In particular, a time series allows one to see what factors influence certain variables from period to period.Time series analysis can be useful to see how a given asset, security, or economic variable changes over time. This helps in judging the future in the context of its past performance.

A data $v[1], v[2], .., v[k-1]$ is likely to be a time series if $v[k]$ depends on the past data. Though this gives an intuition that the memory is long term, i.e, $v[k]$, for any k, strongly depends on $v[1]$ too, this need not always be the case. Some times data could also depend on current and past residuals. The effect could also be dying, in other words, could get weaker and weaker with the more and more past data. An example of this would be weather, where the present weather is strongly dependent on yesterday's but weakly on the weather two months before.

A. Stationarity

A process is said to be stationary if the statistical properties of the process are invariant with time.

Strictly,

$$f(v[1], .., v[k]) = f(v[T+1], .., v[T+N]) \ \forall N, T \in \mathcal{Z}^+$$

where f is probability density function. Often strict stationary is difficult in practice, so a weaker sense of stationary could be defined as invariance upto second order moments. Mean of the process is independent of time and the process has finite variance and is independent of time. There are few additional conditions such as Auto-Covariance should be a function of only Time Difference (Lag) which we will discuss below.

Autocovariance is nothing but a covariance matrix of the data itself considering each sample instance. For example, consider two instances, $v[k]$ and $v[k+l]$, autocovariance $\sigma_{k,k+l}$ is $var(v[k], v[k+l])$. In order to calculate auto-covariance in strict sense, we would need the distribution of each $v[k]$. But when we assume weak stationarity condition, we consider $\sigma_{k,k+l}$ to be independent of sampling time k and hence this allows to compute the variance by considering all possible pairs with time difference(lag) l.

Since the above considers variance, it is also common and convenient to use correlation in the above place. This is hence known as Auto-Correlation Function.

B. Stationary Process

Any signal $v[k]$ can be expanded as $\hat{v}[k] + e[k]$, where $\hat{v}[k]$ is the predictable component and $e[k]$ is noise. In an ideal model, $e[k]$ should be unpredictable or in other words, uncorrelated to other sampling instants. So the ACF (Auto

Correlation Function) is expected to be 1 only at lag 0 (i.e with itself) and is 0 at any non-zero lag.

$$\rho_{ee}[l] = \begin{cases} 1 & \text{if l=0} \\ 0 & \text{if } 1 \neq 0 \end{cases}$$

1) MA Process: When the current prediction depends on the noise/unpredictable component of current instance and of the past instance, the process is said to follow MA (Moving Average) Process assuming (weak) stationarity is held true. An MA process of order 1 depends on the noise component of previous instance, while on order M depends on M such instances.

$$\text{MA(M)} \quad v[k] = \sum_{i=1}^{M} c_i e[k-i] + e[k]$$

We have already seen that the ACF of $e[k]$, the noise component is 1 only at lag 0 and 0 otherwise. Here, since a $v[k]$ depends on past M noise components, the ACF of $v[k]$ zeroes after M instances. (i.e) $\rho[l] = 0$, $\forall l > M$ (abrupt zeroing of ACF after M instances).

2) AR Process: When the present data depends on purely on the past instance, the process when (weak) stationarity is obeyed, is said to be a AR (Auto-Regressive) Process.

An AR process of order 1 depends only on its immediate past, while an AR process of order P depends upto p instances.

$$\text{AR(P)} \quad v[k] = \sum_{j=1}^{P} d_j v[k-j] + e[k]$$

where $e[k]$ is noise/unpredictable component.

Unlike MA process,since it depends on past data, ACF no longer zeroes abruptly but rather decays exponentially. We have seen that ACF is nothing but correlation with the same variable at different sampling instants. In order to establish a similar measure, we try to take the conditional correlation so that link between $v[k]$ and $v[k-l]$ is broken. This is called Partial Auto-Correlation Function (PACF). PACF goes to zero after l=p instances and for a AR Process, the ACF decays exponentially.

The important aspect of this ACF and PACF graph is that they help up in understanding the type of process and the mathematical model behind it which is not known to us before hand in practice. By analysing the nature of ACF and PACF plots, it is possible to come up with a linear time series model.

3) ARMA Process: In practice, a time series model could be a combination of both MA and AR process , (i.e) current predictions could depend on both past data and past residuals. This is collectively called an ARMA process.

$$v[k] = \sum_{i=1}^{M} c_i e[k-i] + \sum_{j=1}^{P} d_j v[k-j] + e[k]$$

Here, both ACF and PACF plots decays exponentially and it's difficult to judge the order of the process without further analysis.

C. Non-Stationary Process

Often, the (weak) stationarity condition may seldom hold. Mean and Variance could change with time. An example of this would be $v[k] = v[k-1] + e[k] + c$, where $e[k]$ is noise. These type tend to accumulate with time and grow upwards. Hence they can not be classified under ARMA process. However, the difference $v[k] - v[k-1]$ is clearly a MA/AR process of order 0. Here, on taking the difference a single time, the process reduces to a stationary process and hence is said to Integrating of order 1. If D differences are needed to reduce a process to be stationary, the process is said to be integrating of order D.

1) ARIMA Process: Let us represent one-time differencing $v[k] - v[k-1]$ as $\nabla v[k]$

$$\nabla^D v[k] = w[k]$$

where w[k] is ARMA process of order (P,M) after D- such differencing. Such v[k]'s are said to in ARIMA of order (P,D,M) . In other words, if repeated D differences reduce a sample to an ARMA process of order (P,M), the sample itself is said to be in ARIMA of (P,D,M).

The presence of integrating effects could also be statistically verified by tests such as *adf, pp, kpss* whose aspects we will not delve into in this work.

D. Relationship with Machine Learning

After all these analysis, this could make one think what this has got to do with concepts learnt in this course. Once a model is determined, the problem reduces to a simple Linear Regression problem of finding the coefficients. For example, let's say a given process is an AR process of order 2. (i.e) AR(P) $v[k] = d_1 v[k-1] + d_2 v[k-2] + e[k]$. Once this is known (by visualizing ACF and PACF plots), the problem reduces to finding the coefficients d_1 and d_2 which could be easily found out by setting the feature matrix X appropriately and using the methods learnt in the first paper to determine the coefficients.

III. The Problem

In this section, We will analyse the stockmarket dataset of different stocks and try to find a low risk stock with comparable returns. We then try to get a forecast of the desired stock that could help making a better informed choice.

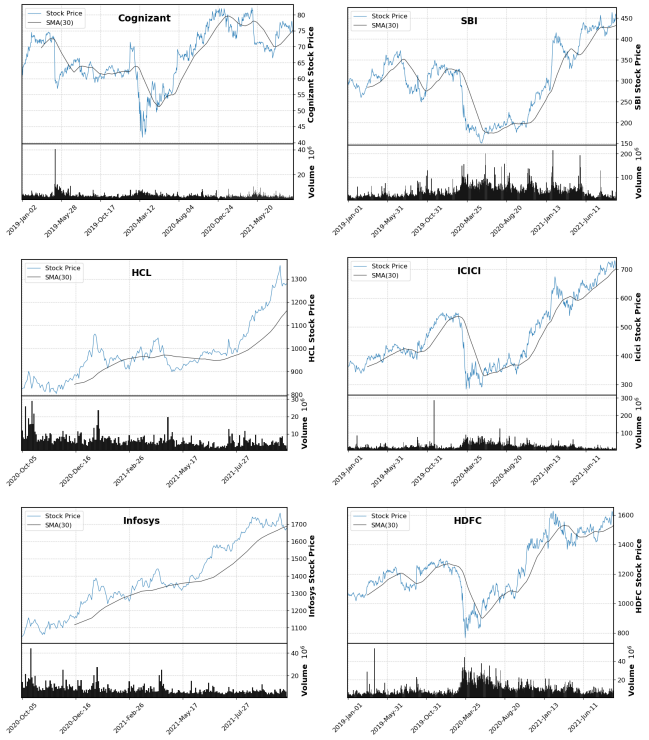
The dataset consists of Open, Low, High and Close prices of six different stocks along with Volume traded. We are also presented with the information of how INR values with respect to USD which could be used to account for inflation and judging if it's worthwhile to invest in stocks in long term at all. Let's start by understanding few finance terms.

Opening Price is the price of a stock at the time of open. This need not be identical to the previous day's closing price.

High Price is the highest selling price of a particular stock in that day. *Low Price* is the lowest selling price of a particular stock in that day. *Closing Price* is the price of a stock at the closing time of market hours. The Normal opening and closing time are from 9.15 a.m. – 3.30 p.m in India. *Adjusted Closing Price* is the adjusted stock's closing price to reflect that stock's value after accounting for any corporate actions post market time.

In a short term trading strategy, all these metrics would be important as these tell the traders about the volatility and the nature of stock from time to time throughout the day. Various plots such as Candle-stick plots, OHLC bar charts helps in visualization to gain insights using such strategies. But this strategy comes with its down-folds such as higher risks and extended real time analysis throughout the day.

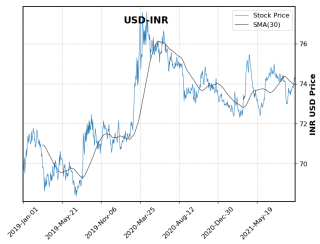
In long term keep and hold strategy, investors would like to spend in a low risk and high returns stock. Typically analysing volatility throughout a day is not necessary for such investments and one would like to keep it simple by tracking only the closing prices over time with the help of line charts.Because line charts usually only show closing prices, they reduce noise from less critical times in the trading day, such as the open, high, and low prices. In this discussion, we'll analyse stocks in a long term perspective and will account Closing Price more into account than other prices.



The blue line represents the stock prices while the black line is the simple moving average (SMA) of 30 days of the stock data. SMA is simply the mean of stock prices for the past 30 days. This helps in filtering random price fluctuations and smoothen it out in order to see the average value. These are used to identify trends and confirm reversals.

1) When price is above the moving average line, we consider the instrument/stock to be in an uptrend and the converse.
2) Breaking of moving average line usually implies trend reversal

For example, one could see a trend reversal around May followed by an uptrend on HCL stocks since May. On all the bank based stocks, SBI, ICICI, HDFC, one could see a downtrend on the March 2020 period which on further investigation reveals that's due to the 'synchronised slowdown' in the World Economy. We could also see a trend reversal post August period implying revertion to normal trends.

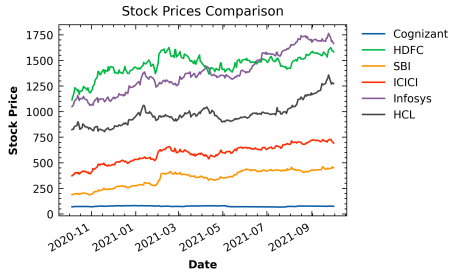


The INR plot confirms our observation that stock prices indeed went down due to decrease in INR value during that time. A trend restoration was seen at similar instances between these plots.

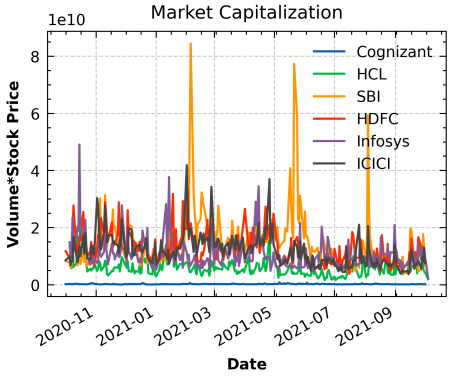
Note that since SMA is based on past prices, they are not ideal for future prices but rather confirms when a trend change has taken place. When the price crosses up and over the SMA, traders take this as a signal to validate their buying.

A. *Comparison between stocks*

Note that the time frame for few stocks are only from 2020-Oct while others are from 2019-Jan. It is worthwhile to look into the common timeframes in the same scale to get a clear idea on which stock performs better.



Most stocks except Cognizant seems to have significant uptrend and in first glance, HCL and Infosys seems to have a steady growth. HDFC seems to have uptrend but shows little stagnation in the recent times. Though these are representative of how the companies does, we should also consider Volume into account to get an estimate of amount obtained through shares.



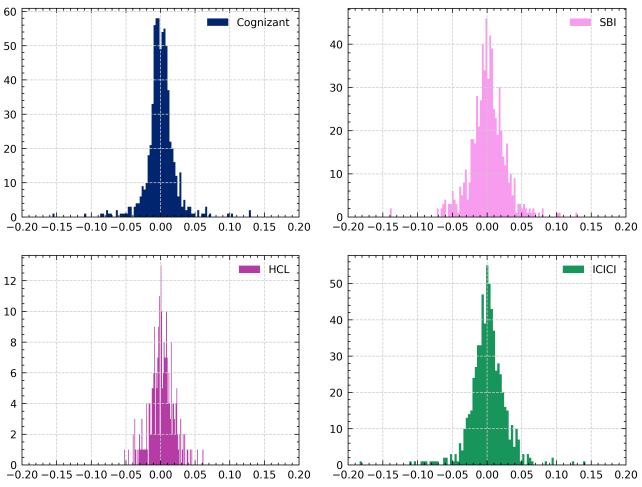
Though in the previous plot the price of SBI shares weren't the highest, SBI has the highest market capitalization. It is then followed closely by Infosys and the rest. One could see that SBI gains the most due to more stock volumes traded which wasn't considered in the previous analysis. Cognizant seems to be the least among these too

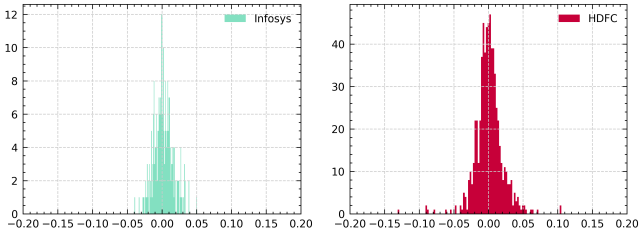
The main factor of importance to any investors is the Rate of return. One would like to earn significant profits per rupee invested, considering for most individuals investments is limited by their financial potential. If $v[k]$ is the cost of stock at k^{th} day, Rate of Return per rupee invested could be expressed as:

$$\text{ROR} = \frac{v[k] - v[k-1]}{v[k-1]} = \frac{v[k]}{v[k-1]} - 1$$

But since safer options such as Fixed deposits (which are almost equal to the inflation rates) are potential options, it is useful to analyse rate of return considering changing value of rupees. If $r[k]$ is the value of INR at k^{th} day,

$$\text{ROR} = \frac{\frac{v[k]}{r[k]}}{\frac{v[k-1]}{r[k-1]}} - 1$$



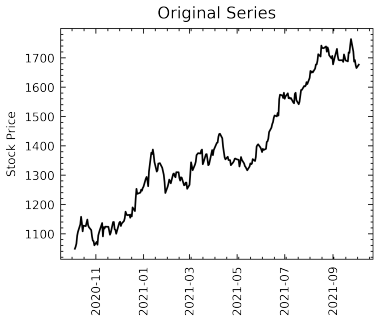


The following charts show the Return of Investments per rupee invested accounting for change in INR value. The plot seems to closely follow normal distribution centred closely to zero. The width of the distribution is representative of the volatility of the stock. Assuming we are looking for a low risk stock, Infosys seems a good option. The below table mentions the Mean returns per day per rupee invested and the associated risk (standard deviation) of the stock analysed in a year period.

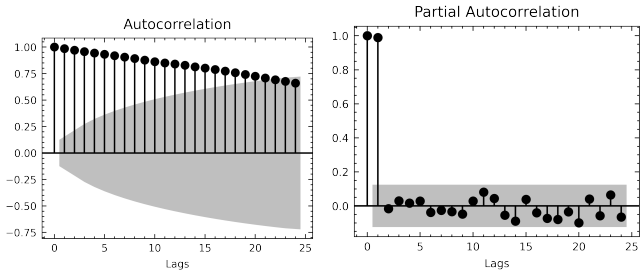
Stock Analysis (Return of Investments)		
Stock	Mean Returns(10^{-3})	Standard Deviation
Infosys	1.9056	0.0138
Cognizant	0.4756	0.0217
HCL	1.8365	0.0172
SBI	0.6150	0.0198
ICICI	0.9952	0.0255
Infosys	1.1023	0.4737

Infosys has the highest mean return with lowest risk. Though the standard deviation appears to be significant, we can expect it to be lower considering we are holding the stock for a long period of time. (Recall, $\hat{\sigma} = \sigma/\sqrt{n}$ and mean per day would remain more or less the same). Considering a risk-averse situation, we'll go ahead and choose this stock and try to forecast using Time Series Modelling. Note that the ideal selection of stock would vary on various factors including risk tolerance of a person, how stocks in a portfolio reacts to the same market situation, corporate announcements and many such factors. We are going ahead with the Infosys stock based on the fact that it has low risk and high returns in the given period analysed.

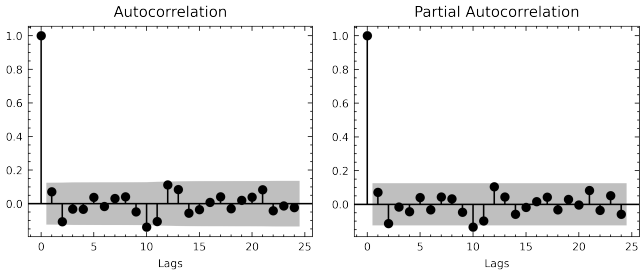
B. Time Series Modelling



On Visualizing the stock, one could see that the stock prices are not stationary and posses some integrating effect. Analysing ACF and PACF plots, we could see that



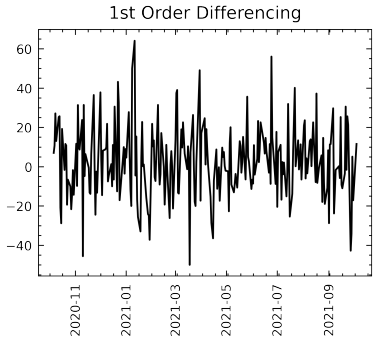
ACF plots shows long memory trend and PACF suggests a strong relationship between $v[k]$ and $v[k-1]$. This means the following could be integrating of order 1. The following was then confirmed with ADF test that it is indeed integrating of order 1. On differencing,



Of the differenced series, ACF and PACF clearly shows zero MA and AR processes. Hence

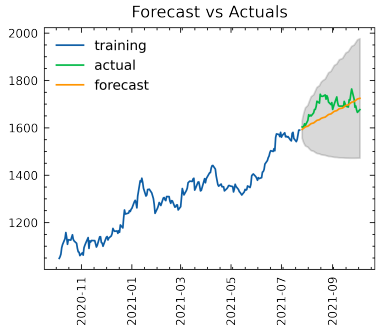
$$v[k] = v[k-1] + e[k] + c$$

is hence the modelling choice of our Infosys stock. The residuals of the differenced series shows white nose/unpredictable like characteristics and thus further strengthens our model assumption.

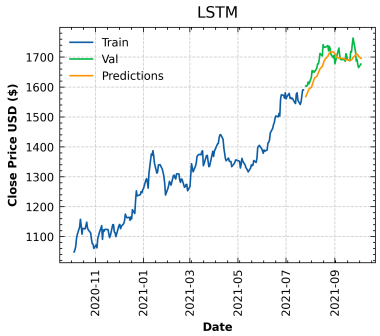


C. Forecast

A Time Series model was analysed and fitted till Oct-2020. The remaining part was forecasted and compared with the true stock predictions.



Through statistical analysis, we are also able to obtain confidence intervals for predictions. Though the model captures the necessary upwards trend and presents a good idea of error estimates, the model seems to relatively simple. A more sophisticated approach would be to use Recurrent Neural Networks to capture patterns with architectures such as LSTM's (Long-Short Term Memory). This will yield much closer results compared to traditional time series analysis and could capture non-linearity in a better fashion.



The above plot shows model predictions by a LSTM model trained and forecasted on the same data. This neural network clearly outperforms our ARIMA model. But due to complex nature and less relevance in the course-content, the mathematics behind such approach were not explored.

IV. Conclusions

Based on our analysis, the key takeaways we had from the following course are:

1) A great amount of information could be obtained by studying and exploring data. It is often essential to know more about the dataset to gain valuable insights rather than just feeding data through algorithms.
2) Real life scenarios such as stock market analysis could be cleverly constructed as Time Series through concepts learned through Linear Regression and Hypothesis testing.

Several additional factors such as following recent trends, having a portfolio reacting to different trends to minimize risks and effective trading strategy could come a long way to improve financial strength of a candidate.

REFERENCES

[1] Principles of System Identification: Theory and Practice, Arun K. Tangirala.

[2] Christopher M. Bishop, Pattern Recognition and Machine Learning

[3] An Introduction to Statistical Learning, Gareth James et al.

[4] Aurelian Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems.

A Mathematical Essay on Linear Regression

Vignesh Kumar S
Department of Chemical Engineering
Indian Institute of Technology, Madras
Email address: ch18b118@smail.iitm.ac.in

***Abstract*—The nexus between incomes and health outcomes is a necessary measure that needs to be established to identify population groups to help with better prognosis. In this work, we examine whether low-income groups are at a greater risk of being diagnosed and dying from cancer. Due to complex nature of the problem and abstractness involved, it is difficult to obtain a good solution using traditional and manual approaches. Adopting a Machine Learning Technique can help in getting insights, analyse different features and large amount of data. The primary goal of this paper is to understand Linear Regression and the Mathematics behind it, use it for modelling and its applicability in real-world scenarios.**

I. Introduction

In this paper, we will study Linear Regression, a technique used for modelling a response variable with one or more explanatory variables in a linear fashion. Then it is used to analyse whether the incidence and cancer mortality occurrences are affected by socio-economic status in the United States. This analysis would aid in obtaining insights among a large population of varying segments and could be used to make decisions involving fundraising and legislation towards this particular cause.

Regression is a technique that attempts to establish a relationship between a target variable and one or more explanatory variables. Linear Regression is a regression method that models the relationship linearly. The best fit line obtained by minimizing the sum squared distance between the model predictions and the outputs is essentially the idea behind this technique. This could also be used to establish if there is any correlation between variables. However, this does not imply the output is the cause of input features but is just a technique to see if there is some relationship between them. Linear Regression is also widely used in time series analysis such as stock market predictions, weather conditions, sales in a particular store.

The dataset used for this problem comprises cancer incidences and mortality across various areas in the United States and the median income split among ethnicities in each area. The data also contains the number of people with health insurance and the trend of cancer incidence in the past 5 years. We use Linear Regression to demonstrate whether cancer incidence and mortality are strongly correlated with socio-economic status. The insights obtained from this could be used by firms and the government to make decisions and solve issues.

In this paper, we will study the concepts behind Linear Regression and its various methods. We then use this technique to establish the presence of a relationship between cancer incidence and socio-economic factors such as income, poverty, and the percentage of people with access to insurance.

II. Linear Regression

Regression is a statistical technique where both the dependent and independent variable takes continuous values and a model is fitted on the explanatory variables. If the model that is learnt is linear, this is called a linear regression. Methods of estimation of the parameters could be done in several ways including Least Squares and statistical techniques such as Maximum Likelihood Estimation, Bayesian Estimation. A few extensions of it also include Ridge Regression, Lasso, Elastic Net and Online based learning such as Recursive Least Squares method. The choice of the model is based on the A priori knowledge of the process, observation and visualization of the data and the fit of the model.

Least Squares Estimators

Let the number of datapoints be N and k be a integer such that $1 \leq k \leq N$ and be used to represent a particular instant in the datapoint. Ordinary least squares problem deals with finding the best prediction of $\vec{y}$ using p explanatory variables (or) regressors $\psi_i[k]$, i = 1,2,...p such that ŷ[k] are collectively at a minimum distance. When the predictor are linear in parameters, we have linear least squares problem.

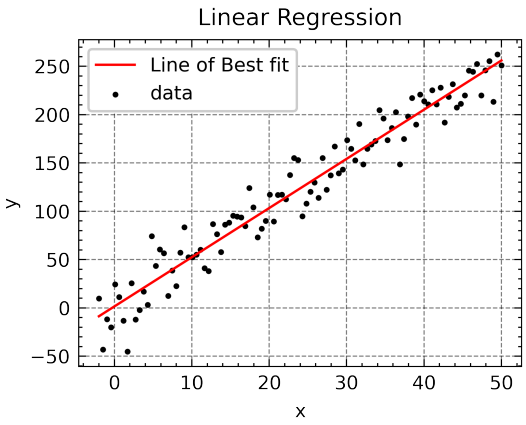


Fig. 1: Example of Linear Regression

The General form of a linear regression model is

$$\hat{y}[k] = \sum_{i=1}^{p} \psi_i[k]\theta_i \tag{1}$$

$$\varepsilon = \vec{y} - \hat{y} \tag{2}$$

where ε is the residual vector.

A. Ordinary Least Squares Method

Let's represent the predictions

$$\hat{y}[k] = \sum_{i=1}^{p} \psi_i[k]\theta_i \text{ in a matrix notation}$$

$$\psi[k] = \begin{bmatrix}\psi_1[k] & \psi_2[k] & .... & \psi_p[k]\end{bmatrix}^T$$

$$\text{Let } \Phi = \begin{bmatrix}\vec{\psi}[1]^T \\ \vec{\psi}[2]^T \\ ... \\ \vec{\psi}[N]^T\end{bmatrix}$$

$$\underset{n\times 1}{\mathrm{Y}} = \underset{n\times p}{\Phi} \times \underset{p\times 1}{\theta} + \underset{n\times 1}{\varepsilon}$$

Cost function (or equivalent to negative reward function) is a measure of penalty between the predicted outputs and true outputs. While several choices could be worked on, Ordinary least squares tries to minimize the sum squared error.

Obj Function:

$$\min_{\theta} \quad J_n(\theta) = \sum_{i=1}^{N} (y[k] - \hat{y}[k])^2$$

In matrix form, this could alternatively be expressed as

$$\min_{\theta} \quad J_n(\theta) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$$

where $\hat{y} = \Phi\theta$

On taking the gradient and solving for the optimum minimum point yields the solution,

$$\hat{\theta}^* = (\Phi^T\Phi)^{-1}\Phi^T y$$

Note that a constant term could be easily included in the model by incorporating

$$y[k] = \Psi^T[k]\vec{\theta} + \beta \quad as$$

$$y[k] = \begin{bmatrix}\Psi[k] & 1\end{bmatrix}\begin{bmatrix}\vec{\theta} \\ \beta\end{bmatrix}$$

B. Gradient Descent

Optimization of minimizing the cost function can be done in an alternate way, Gradient Descent. Gradient is the direction along which a function increases the maximum. Going along the opposite direction of Gradient gives the maximum decrease. On each iteration, the cost is reduced by moving along this direction till a certain step size, which is influenced by the hyper parameter learning rate (α). This is continued till a minima is reached or a stopping criterion is met.

1) Parameter Initialization: Parameters can be assigned random values to begin with. In linear regression, often, the parameters are initialized to zero.

2) Stopping Criteria: The convergence to local minima is checked by evaluating if a stopping criterion is met. Some of the choices for stopping critera include:

- Change in cost function is less than a threshold
- Change in gradient is less than a threshold
- Change in parameters is less than a threshold
- Number of iterations has reached its threshold

ALGORITHM: Initialize θ
Repeat till convergence
{
for $j \in \{1, 2, .., p\}$
$\theta_j := \theta_j - \alpha \frac{\partial J(\theta_1, \theta_2, ..., \theta_p)}{\partial \theta_j}$
}
where α is the learning rate.

3) Effect of Learning Rate: Small Values of α may take too many iterations to reach minima. In the algorithm above, small values of α barely updates the model parameters while a large α produce significant changes. This means that very small values will take too many iterations and if α is large enough, it may oscillate and even diverge from the optimum point.

This itself could be set up as an additional optimization problem to obtain the best α at each iteration to converge much faster. However, in practice, α is held constant and is set at appropriate levels which ensures convergence at a reasonable rate.

Though gradient descent for a random function with multiple local minimas may end up in different convergence, but for a Least squares objective function J, we always reach the global minima provided the learning rate is set at appropriate levels.

Gradient descent could be of multiple types. Ordinary, Stochastic, Mini-Batch, Batch e.t.c. These are not discussed as it is not under the scope of this paper and could be taken up as an extensive study under optimization course.

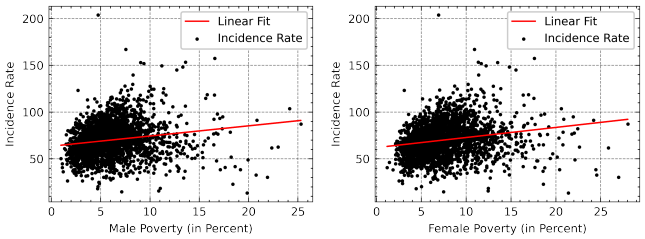
III. The Problem

In this section, We will do a case study on whether socio-economic factors such as income, poverty, subscription to health insurance is related to incidence and mortality of cancer. Linear Regression technique is used to understand our plots better and strengthen our claim.

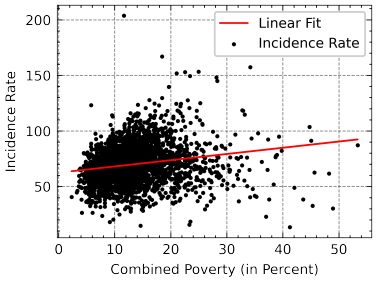
A. Poverty

We attempt to gain insights by visualizing each socio-economic factors versus incidence and mortality rates separately. With the help of linear regression, we try to see if there is a trend associated with the factors.

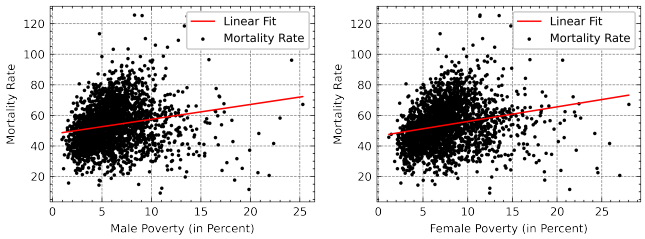
First, let us visualize whether if there is any correlation between Poverty versus Incidence and Mortality Rates. Access



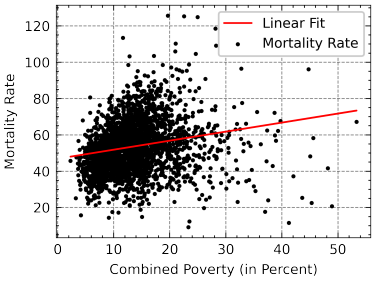
(a) (b)



(c)



(a) (b)



(c)

to good healthcare and proper nutrition is difficult for people living under poverty. In a country like US, where health care is not free unlike UK, by intuition, this seems to have a direct impact on the disease prognosis. The absolute count of people living in poverty may not be a very useful measure. Plots are constructed between Poverty Percentage versus Target Variables where poverty percentage is defined as count of people living under poverty in an area by estimated population (in percent)[1]of that area.

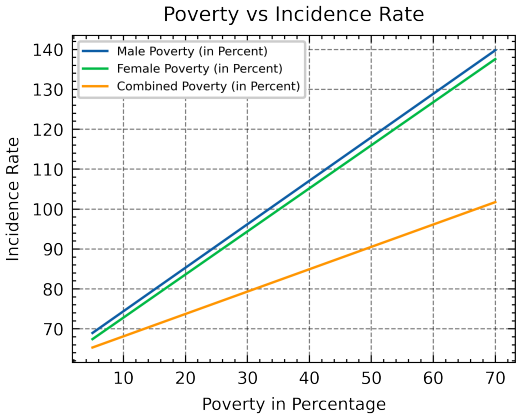


Fig. 2: Influence of Poverty versus Cancer Incidence

Fig.2 strengthens the evidence that as percentage of people living in poverty increases, the incidence rate goes up. Male Poverty and Female Poverty show similar trends implying that gender does not impact much on the incidence rate.

[1]Estimated population = $10^6 * \frac{AnnualizedIncidenceRate}{IncidenceRate}$

Annualized Incidence is the absolute count of people diagnosed
Incidence Rate is the count of people diagnosed per million population in an area.

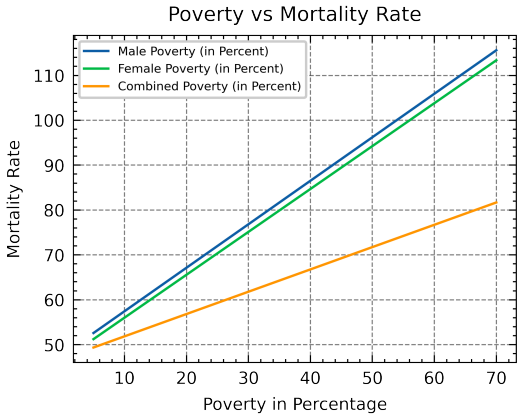


Fig. 3: Influence of Poverty versus Cancer Mortality

Similar trends are oberserved across Poverty versus Mortality Rate. This suggests that people living under poverty are more vulnerable to cancer and related diseases compared to well off people.

B. Income

In this section, we will analyse how incomes impact cancer incidences and Mortality Rates. We will visualize total incomes and incomes based on ethnicities and try to identify if there is any correlation between cancer rates.

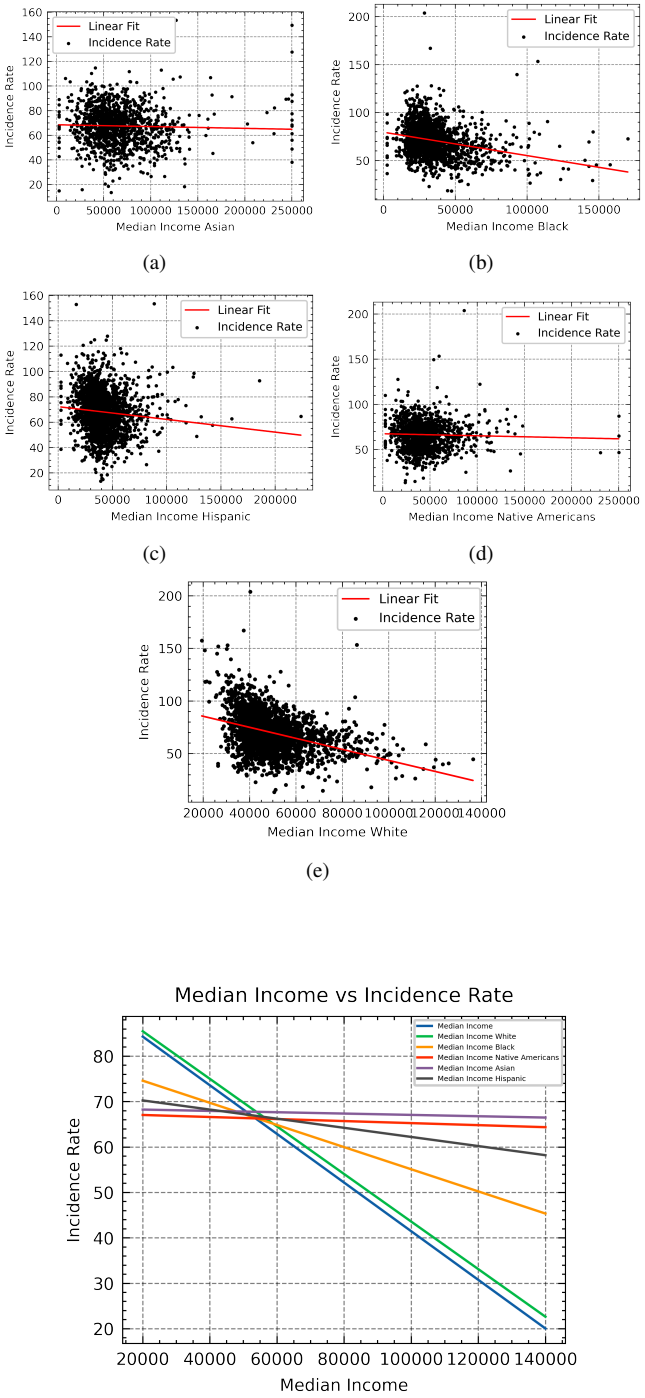


Fig. 4: Influence of Income versus Cancer Incidence

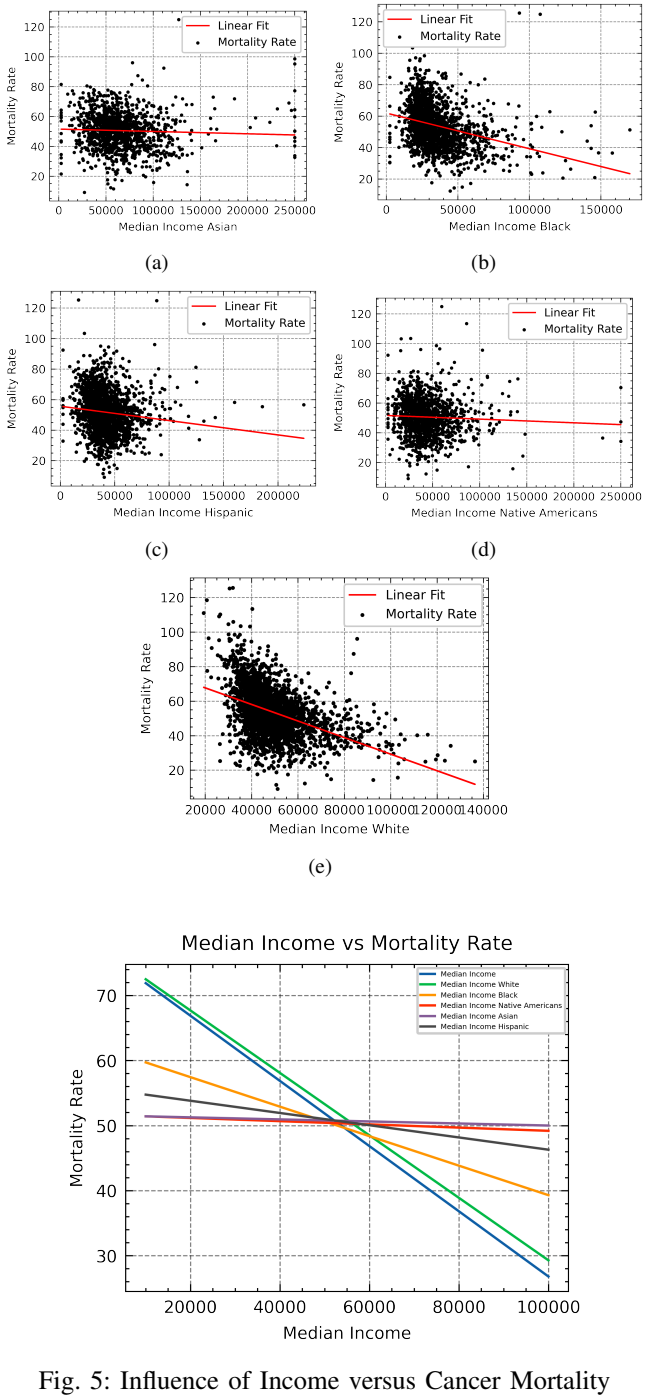


Fig. 5: Influence of Income versus Cancer Mortality

From Fig.4, Income does play a role. People with lower incomes have difficulty in managing health care expenses and obtaining nutritious food. This could also be attributed due to income being closely related with poverty. Hence such trend could have been observed.

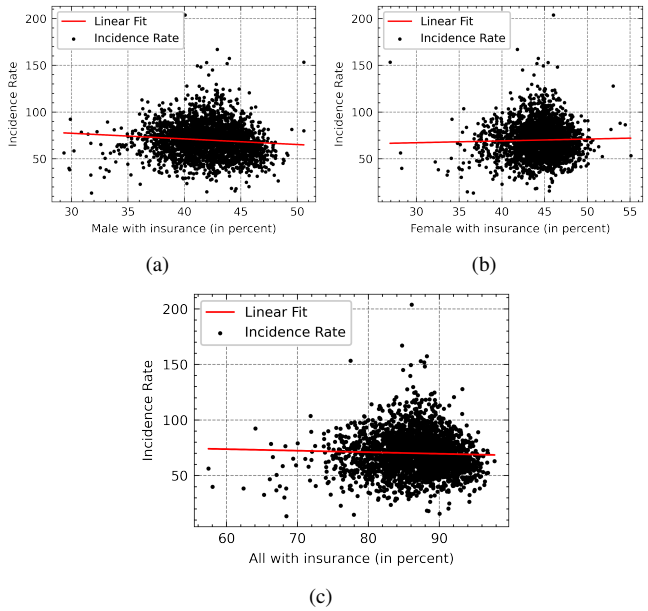
Fig 5. shows mortality rate of lower income people are high and decreases steeply as income increases. This substantiates the fact that people with lower income are more susceptible to cancer.

C. Health Insurance

In this section, we will analyse whether population subscribed to health insurance have an advantage and are less prone to cancer incidence and mortality. We will look across the effects across gender and population as a whole and attempt to interpret the results.

The absolute count of people with Health Insurance may not a useful measure. So plots are constructed for Percentage of people with health insurance versus the Target Variables, Incidences and Mortality Rates. Percentages are number of

people with Health Insurance per estimated population of an area[2]



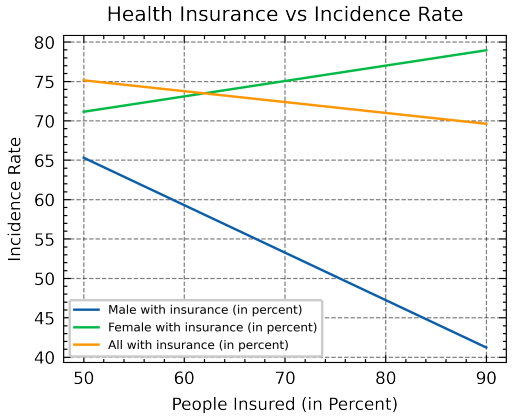
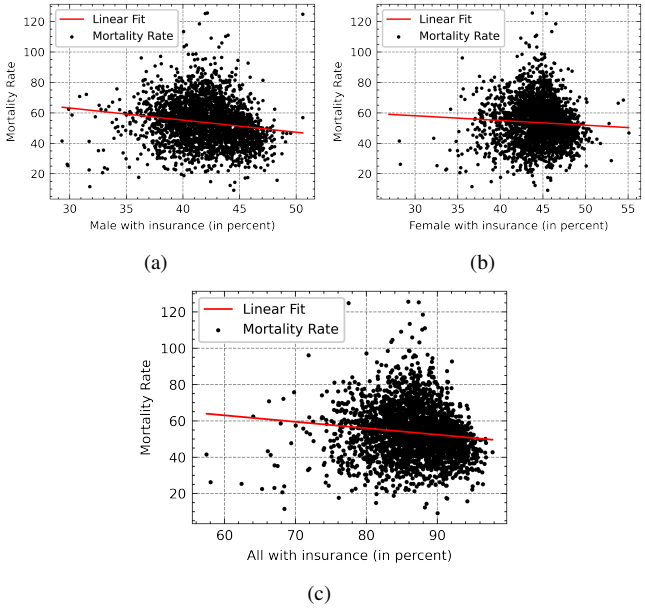


Fig. 6: Influence of Health Insurance versus Cancer Incidence

From Fig.6, The incidences corresponding to areas with All percentage of people with high insurances are almost the same or a little lower than those with low insurance percentages. This makes sense because Insurances are usually availed after a medical treatment and doesn't influence much on the probability of getting cancers. The slight decrease in nature could be speculated due to relatively more number of people subscribing to health insurance who are above poverty line. But, this cannot be justified as people below poverty line also tend to resort to health insurance in case of emergencies. In fact, Incidences of Females with insurance slightly increase (or remain almost constant) with increase in insurance percentage. Hence, it remains inconclusive whether Health Insurance are really effective in reducing the incidence rates.



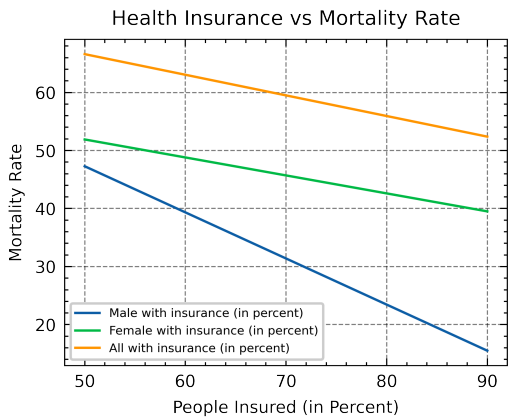


Fig. 7: Influence of Health Insurance versus Cancer Mortality

From Fig.7, Mortality Rate shows down trending values as percentage of people insured increases. This makes sense that people would access health care without the intimidation of financial burden and the chances of survival increases. The findings varies from our previous conclusion on incidence because the probability of getting diagnosed with cancer may not be influenced by Health Insurance but the survival would be improved due to medical facilities and less financial burden.

D. Statistical Evidence

In statistics, the p-value is the probability of obtaining results at least as extreme as the observed results of a statistical hypothesis test, assuming that the null hypothesis is correct. A smaller p-value means that there is stronger evidence in favor of the alternative hypothesis. In a linear regression, the

[2]Estimated Population ≈ People with Insurance + People without Insurance.
The Previous estimate of population[1] could also be used here. In fact, both the estimates are very similar and small differences might be due to census errors

alternate hypothesis being existence of relationship between response and dependent variables, lower the p values, the more significant is that feature in the model.

F-statistic is an indicator of whether there is a relationship between dependent variable and the response variables. Because individual t-tests assume that each variable comes from an independent distribution which is not always the case. This leads to multi-collinearity issues and also accumulation of Type-1 error to falsely conclude the results. Hence, by judging by the F-stat scores, we could conclude if there is a significant relationship between dependent variables and the response variable. The larger the F statistic is away from 1 the better it is.

R-Squared is a statistical measure of fit that indicates how much variation of a dependent variable is explained by the independent variable(s) in a regression model. The closer the R-squared is to 1, the more likely the relationship is statistically linear.

TABLE I: All Poverty Percentage

Metric	Incidence Rate	Mortality Rate
F-stat	99.89	124.6
P-val const	0.001	0.000
P-val feature	0.001	0.000
R^2	0.036	0.045

TABLE II: Median Income

Metric	Incidence Rate	Mortality Rate
F-stat	444.8	652.2
P-val const	0.000	0.000
P-val feature	0.000	0.000
R^2	0.144	0.198

All models except Insurance Percentage vs Incidence Rate have high F-statistic value implying the presence of some relationship between dependent and predictor variables. Though the R-squared value is less which suggests that the actual relationship could be non-linear, one can nevertheless see that pvalues of features are less than 0.05 strengthening the conclusions made in the above section.

TABLE III: Insurance Percentage

Metric	Incidence Rate	Mortality Rate
F-stat	3.903	41.26
P-val const	0.001	0.000
P-val feature	0.058	0.000
R^2	0.001	0.015

For the model between Insurance Percentage vs Incidence Rate, the pvalue lies slightly above 0.05 indicating that the linear regression model may not be statistically significant. In fact, both the F-statistic and the R-squared metrics are among the lowest indicating that the linear model would be inconclusive. This suggests our claim that Health Insurance doesn't have statistical influence on Incidence Rate as Insurances are usually availed after a medical treatment and doesn't influence much on the probability of getting cancers.

IV. Conclusions

Socio-Economic factors such as incomes, poverty and access to health insurance does influence incidence and Mortality Rates. High incomes and Low poverty helps in preventing the disease and also gives more chances of survival due to better access to health care systems. Though Health Insurance is not very effective in preventing cancer incidence, it reduces the Mortality rate and increases the chance of survival.

Targeted programs towards population with low income and regions of high poverty must be planned out to help prevent cancer mortalities. Incentives on Health Insurance plans and decreased hospital charges towards these section of people could be planned to reduce the incidence and the effects of cancer.

Possible avenues of research could include analysing percentage of people having access to good health care system, the reason for not pursuing health insurance which could be, but not limited to, lack of wide range of plans catering to their capability.

References

[1] Aurelian Geron, Hands-On Machine Learning with Scikit-Learn,Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, pp. 38–111

[2] Arun K. Tangirala, Principles of System Identification, Theory and Practice

A Mathematical Essay on Logistic Regression

Vignesh Kumar S
Department of Chemical Engineering
Indian Institute of Technology, Madras
Email address: ch18b118@smail.iitm.ac.in

***Abstract*—The purpose of this article is to understand Logistic Regression and the mathematics behind it. In this work, we also demonstrate the application of logistic methods illustrated through one of the most infamous shipwrecks in history, the sinking of the Titanic. Due to the complex nature and the abstractness involved in predicting the survival of a candidate, it is often difficult to obtain a good solution through manual approaches. Adopting a machine learning technique can help in obtaining insights and analyse different features in a large amount of data comprehensively.**

I. Introduction

In this paper, we will study Logistic Regression, a technique used to analyze and model the probability of a certain class, usually a dichotomous outcome. We will use this technique to analyze whether the survival of passengers was related to factors such as age, affordability, gender, travelling along with a group. This analysis would aid in obtaining insights among a large population and understand the trend behind survival and biases that could have existed at that time.

The classification problem attempts to establish a relationship between a categorical target variable with one or more explanatory variable(s). Logistic Regression is a discriminatory statistical modelling technique, which itself models the probability of outputs in terms of inputs. This in general, is well suited for describing and testing hypotheses about relationships. A Binary classifier is constructed by choosing a threshold and classifying inputs with greater probability as one class and below cutoff as another class.

The dataset used for this problem comprises of names of the passenger along with their titles, age, whether they travelled along with siblings or spouses, parents or children. It also contains ticket class, fare, port of embarkation, cabin number. We use logistic regression to learn and predict the survivability of passengers given these input features.

This work represents the concepts behind Logistic Regression and evaluation metrics involved. We then use this technique to establish the relationship to predict the survivability of passengers using passenger data.

II. Logistic Regression

The goal in classification problem is to take an input feature $\boldsymbol{x}$ and assign it to one of the two classes (K classes in general). The input space is thereby divided into decision regions whose boundaries are called decision surfaces. Logistic Regression is a linear model for classification, which means the decision surfaces are linear functions of the input vector $\boldsymbol{x}$.

The simplest case is to model $y(\boldsymbol{x}) = \mathrm{w}^T\boldsymbol{x} + w_0$ so that y is a real number. But for a classification problem, we seek to obtain probabilities that lie in (0,1) range. Hence, we apply a non-linear function $f()$ such that

$$y(\boldsymbol{x}) = f(w^T\boldsymbol{x} + w_0)$$

returns the posterior probabilities. Decision surfaces correspond to $y(\boldsymbol{x}) = constant$, which implies $w^T\boldsymbol{x} + w_0 = constant$ for one to one f.

The non-linear function $f()$ used in logistic regression is a sigmoid function that outputs a number between 0 and 1.

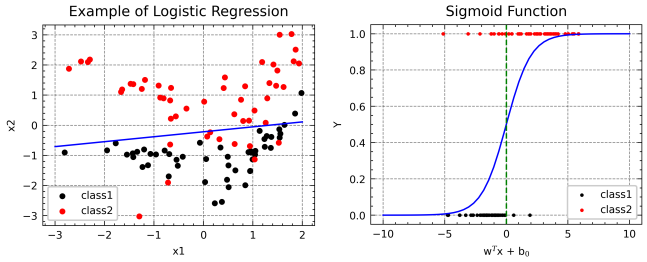
$$\sigma(u) = \frac{1}{1 + e^{-u}}$$

$$\hat{p} = \sigma(w^T\boldsymbol{x} + w_0)$$

The estimated probabilities can easily be converted into a binary classifier by predicting

$$\hat{y} = \begin{cases} C_1 & \text{if } \hat{p} < Threshold \\ C_2 & \text{if } \hat{p} \geq Threshold \end{cases}$$

where C_1 and C_2 are the two classes. The threshold is usually set to 0.5.



(a) Logistic Regression decision surface separating two classes (b) Predicting classes based on Threshold

A. Discriminative Model Setting

Let's try to model the above in a discriminative setting which models $P(Y|X)$, where X, y $\in \mathbb{R}^{\mathrm{d}}$ x $\{\pm 1\}$. Note that we have used y as $\{\pm 1\}$ for the ease of mathematical modelling whereas $\{0,1\}$ could also be used in general. We assume,

$$P(Y = 1|X = \boldsymbol{x}) = \sigma(w^T\boldsymbol{x}) \tag{1}$$

Since we are dealing with a binary classifier now,

$$P(Y = -1|X = \boldsymbol{x}) = 1 - \sigma(w^T \boldsymbol{x}) = \sigma(-w^T \boldsymbol{x}) \quad (2)$$

The bias term w_0 could easily be incorporated in $\boldsymbol{w}$ as an extra feature and $\boldsymbol{x}$ as [$\boldsymbol{x}$,1] whenever required.

Equations (1) and (2) could be combined into a single expression as follows:

$$P(Y = y|X = \boldsymbol{x}) = \sigma(y w^T \boldsymbol{x}) \quad (3)$$

B. Parameter Learning

The parameters can be estimated by Maximum Likelihood estimation. Let the number of data points be *M* and *i* be an integer such that $1 \leq i \leq M$ and be used to represent a particular instant in the datapoint.

$$\begin{aligned} Likelihood\, \mathcal{L}(\boldsymbol{w}) &= P(y_1, y_2, ...y_M|x_1, x_2, ...x_M) \\ &= \prod_{i=1}^{M} P(Y_i = y_i|X_i = xi, \boldsymbol{w}) \\ &= \prod_{i=1}^{M} \sigma(y_i w^T \boldsymbol{x_i}) \end{aligned}$$

$$\log \mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{M} \log \sigma(y_i w^T \boldsymbol{x_i})$$

We could maximise log-likelihood or minimise negative log-likelihood to estimate the parameters.

$$\hat{R}(\boldsymbol{w}) \triangleq -\log \mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{M} \log\left(1 + exp(-y_i w^T \boldsymbol{x_i})\right)$$

where $\hat{R}(\boldsymbol{w})$ is called Empirical logistic loss function. To get the minimum cost, the gradient at optimum should tend to zero.

$$\nabla \hat{R}(\boldsymbol{w}) = \sum_{i=1}^{M} \sigma(-y_i w^T \boldsymbol{x_i})(-y_i \boldsymbol{x_i})$$

However, there are no closed-form solutions available for the above optimization problem. Hence, we resort to Gradient descent to reach the minima.

C. Understanding the Loss function

For the i^{th} datapoint, $w^T \boldsymbol{x_i}$ determines the nature of prediction. If the sign (+ or -) of $w^T \boldsymbol{x_i}$ matches with y_i,the predicted value is correct and the empirical loss function is less. If the predicted value does not match with the true prediction, the cost incurred becomes high. So the loss function maps a high cost to a misclassification point and hence minimising this, we obtain a solution with lower misclassification error.

D. Gradient Descent

The Gradient is the direction along which a function increases the maximum. Going along the opposite direction of Gradient gives the maximum decrease. On each iteration, the cost is reduced by moving along this direction to a certain step size, which is influenced by the hyperparameter learning rate (η). This is continued till a minima is reached or a stopping criterion is met.

1) Parameter Initialization: Parameters can be assigned random values, to begin with. In logistic regression, often, the parameters are initialized to zero.

2) Stopping Criteria: The convergence to local minima is checked by evaluating if a stopping criterion is met. Some of the choices for stopping criteria include:

- Change in the cost function is less than a threshold
- Change in gradient is less than a threshold
- Change in parameters is less than a threshold
- Number of iterations has reached its threshold

ALGORITHM Initialize $\boldsymbol{w_1}$
Repeat till convergence
{

$$\begin{aligned} \boldsymbol{w_{t+1}} &:= \boldsymbol{w_t} - \eta \nabla \hat{R}(\boldsymbol{w_t}) \\ &= \boldsymbol{w_t} + \eta \sum_{i=1}^{M} \sigma(-y_i w^T \boldsymbol{x_i})(-y_i \boldsymbol{x_i}) \end{aligned}$$

}

where η is the learning rate and t is the iteration number.

E. Regularised Logistic Regression

In order for the model to generalise well and to prevent overfitting, a penalty function could be added to the loss function. The choices of penalty function could be $l1$ norm (Lasso), $l2$ norm (Ridge) or a combination of both (Elastic Net). For a $l2$ norm penalty function, Empirical loss function is:

$$\hat{R}(\boldsymbol{w}) = \sum_{i=1}^{M} \log\left(1 + exp(-y_i w^T \boldsymbol{x_i})\right) + \frac{\lambda}{2} \|\mathbf{w}\|_{\mathbf{2}}^{\mathbf{2}}$$

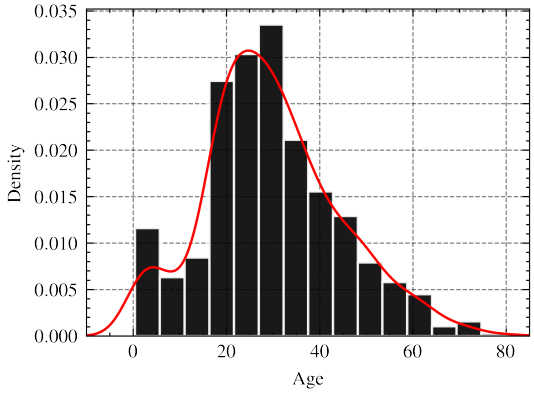
The parameters are obtained through the gradient descent method illustrated earlier.

III. The Problem

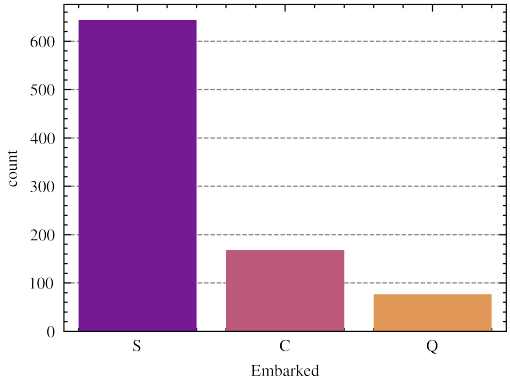
In this section, We will analyse the Titanic dataset and try to predict the survival of passengers based on the information available. The logistic regression technique is used as a classification model in further analysis. Let's start by addressing any potential missing values.

A. Missing Values

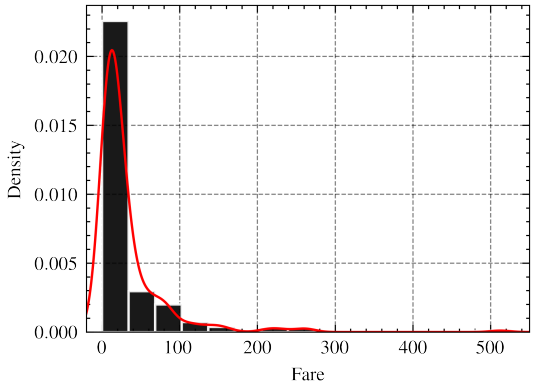
1) Age: Since Age doesn't follow uniform distribution, using constant imputation might not always gives the best results. Let's use linear regression based imputer to fill the missing values.



2) Port of Embarkation: Whenever we encounter missing values for embarked, we impute the missing values with Southampton, the port most people boarded,

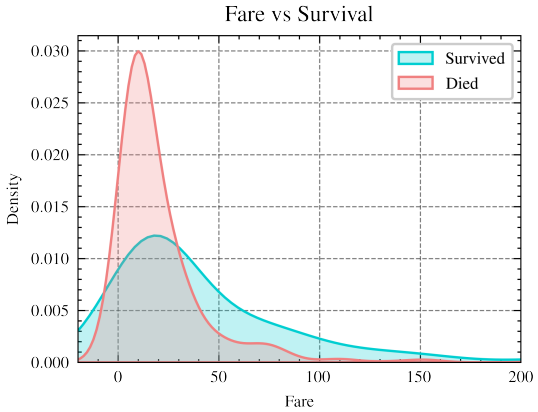


3) Fare: Again following a linear regression based imputation would work well compared to a constant imputation method due to skewed distribution.

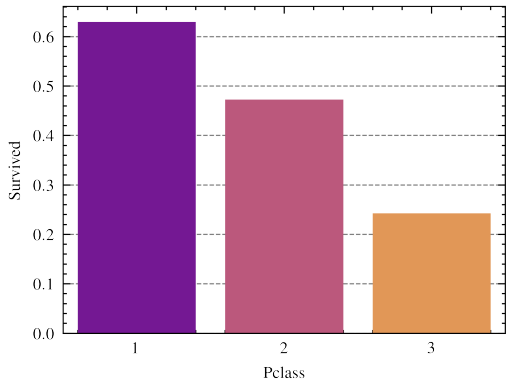


B. Exploratory Data Analysis

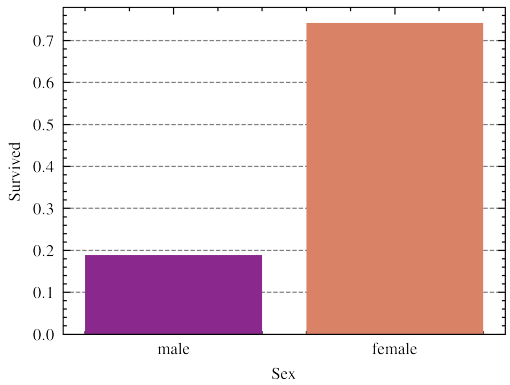
1) Age versus Survival: The distribution of the survived and dead are similar. One notable difference is that a large proportion of children survived. This shows evident attempts were taken to save children by giving them a place on life rafts.



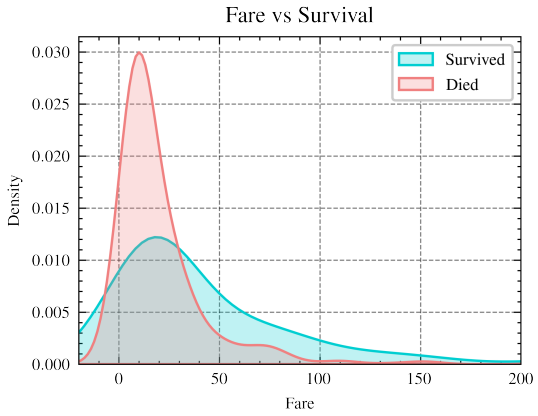
2) Ticket Classes versus Survival: The first class was the safest and the third class was the unsafest travelling option. Since there is a inherant order in the nature of class, let's stick with label encoding for ticket class.



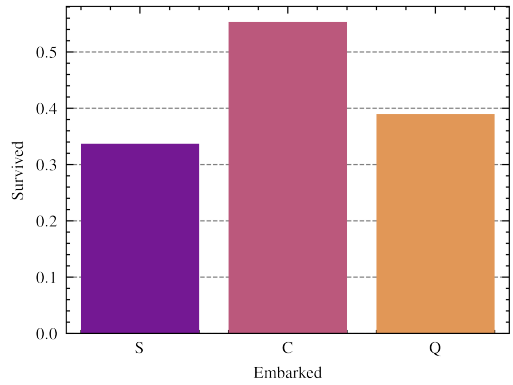
3) Gender vs Survival: Females had more chance of survival. This could be because more importance has been given to them during evacuation (in accordance with the movie).



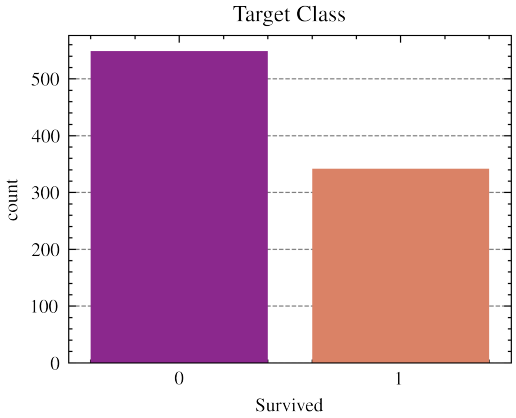
4) Fare vs Survival: Passengers who were able to afford more had a higher survival rate compared to those who couldn't. This could also be related to the location of rooms offered and the ease of access to lifeboats during the tragedy.



5) Embarkation vs Survival: People who boarded from Cherbourg, France have had the highest survival rate, and those who boarded from Southampton were less likely to survive. While this doesn't make much sense intuitively, the average fare spent by a Cherbourg passenger was 59.95 dollars, the highest among all the three locations. This could mean that people boarded from Cherbourg, on average, were significantly richer and influential and had better chances of survival.



6) Imbalancedness: There is a slight imbalancedness in the dataset. There are more passengers who have not survived (61.6%) and less who have survived (38.4%). Though this is not significant, we will grid search if we need to apply any class weights to correct this measure.



C. Feature Engineering

1) Honorifics: The name attribute along with first and last name contained titles. Honorifics (Titles) were quite commonly used during the era in which the tragedy happened. This would be a useful feature to extract which could determine how rich and influential a person was. It is also interesting to see from the training dataset that the captain didn't survive and went down with the ship.

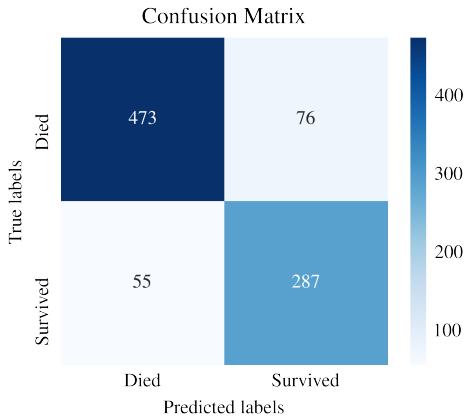
2) Cabin Group: Even though Cabin contained a lot of missing values, considering only the cabin group of the available data and taking missing values as a separate category showed some trends. The cabin values are also representative of the ticket classes and the closeness to lifeboats. Hence, a new feature *Cabin Group* was constructed.

3) Ticket shared: The ticket attribute consists of ticket numbers with or without some added prefix. Without available information, it's difficult to establish a relationship about the prefixes. Dropping prefix, a closer analysis on tickets showed that the tickets id were not unique. Survival rates of passengers could be predicted upon knowing the survival of candidates who had shared the same ticket number. A new attribute was constructed which mapped the number of people who shared common ticket numbers.

One-Hot encoding was used to preprocess data without inherent order and label-coding was used with data that possessed some order (eg. ticket class)

D. Model Evaluation

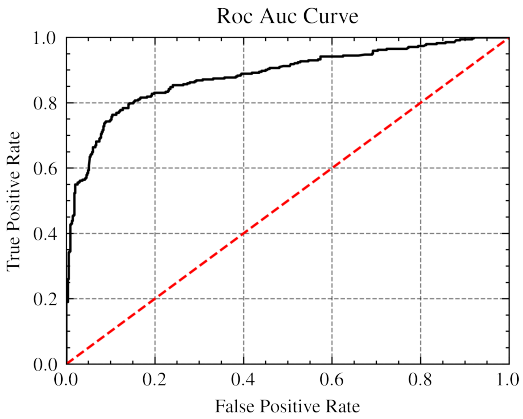
After Data cleaning, Preprocessing and Feature Engineering, Using the concepts learnt through SVM, a kernel based Logistic regression model with Gaussian kernel was trained and tuned over hyper-parameter λ and C. This gave a significant performance boost compared to previous model due to the fact that this enables the model to capture non-linearities better. This could also count due to the improved imputation method used in the new analysis. The confusion matrix and the evaluation metrics for Gaussian Kernel Logistic Regression are reported below:



Kernel Logistic Classifier	
Metric	Score
Accuracy	0.8529
Precision	0.7906
Recall	0.8391
F1 score	0.8141

In a classification setting, accuracy may not be the best score to consider especially in a skewed class situation. The training data in our case is quite balanced and not a concern. We also evaluate the F1 score and the scores of precision and recall. Precision is the ratio of correct positive predictions to the total positive predictions of our model. The recall is the ratio of positive instances that are correctly detected by our classifier. The F1 score is simply a harmonic average of these two. Our classifier seems to have performed well in all the metrics discussed. Both the accuracy and F1 score have improved approximately two percent which suggests an improvement over linear logistic regression model

ROC curve is another common tool used with binary classifiers. The ROC curve plots the true positive rate (another name for recall) against the false positive rate. The FPR is the ratio of negative instances that are incorrectly classified as positive. It is equal to one minus the true negative rate, which is the ratio of negative instances that are correctly classified as negative. The TNR is also called specificity. Hence the ROC curve plots sensitivity (recall) versus 1 – specificity.



The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner). One way to compare classifiers is to measure the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5. The area under the ROC curve of our logistic regression classifier is *0.8833*.

IV. Conclusions

While there was some amount of luck involved, Socio-Economic factors such as incomes, influences, class tickets and factors such as gender, travelling with children seems to have impacted the survival rate. This could also be used to explain the inherent biases and favouritism we hold as a society towards the rich section of population.

Extensions to traditional Logistic Regression like use of Gaussian Kernels and the kernel trick usually applied to SVM's and better imputation gives improvements in the prediction of logistic regression classifiers.

References

[1] Aurelian Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, pp. 112–180
[2] Christopher M. Bishop, Pattern Recognition and Machine Learning
[3] Joanne Peng, An Introduction to Logistic Regression Analysis and Reporting

A Mathematical Essay on Naive Bayes Classifier

Vignesh Kumar S
Department of Chemical Engineering
Indian Institute of Technology, Madras
Email address: ch18b118@smail.iitm.ac.in

***Abstract*—The purpose of this article is to understand the Naive Bayes classifier and the mathematics behind it. In this work, we demonstrate the application of the Bayes classifier illustrated through the 1994 Census bureau database by Ronny Kohavi and Barry Becker. Due to the complex nature and the abstractness involved in determining the income of a person, it is often difficult to obtain a good solution through manual approaches. Adopting a machine learning technique can help in obtaining insights and analysing different features in a large amount of data comprehensively. We try to establish a nexus between income range and different features involving the bio-data of a person.**

I. Introduction

In this paper, we will study Naive Bayes classifier, a technique based on Bayes theorem used to analyze and model the probability of a certain class, either a dichotomous or multiple outcomes. We will use this technique to determine whether a person makes over $50K a year using factors such as age, Work-class, education, marital-status, occupation, race, sex, native country etc. This analysis would aid in obtaining insights among a large population and understanding the trend behind income and various other factors.

The classification problem attempts to establish a relationship between a categorical target variable with one or more explanatory variable(s). Naive Bayes Classifier is a generative statistical modelling technique that can be considered as fitting a probabilistic model that optimises the joint likelihood $p(C, x)$. Conditional probability of outputs in terms of inputs can equivalently be constructed which can then be used to infer the predictions with certain assumptions on likelihood. They are one of the simplest Bayesian models but with density estimation and on certain datasets, they can work well and achieve high levels of accuracy.

The dataset used for this problem comprises age, working class, number of working hours per week, occupation, education levels, capital gain and loss. It also contains marital-status, relationship, race, sex and Native country. We use Naive Bayes classifier to learn and predict whether a given individual earns above $50K dollars a year given these input features.

This work represents the concepts behind Naive Bayes classifier and evaluation metrics involved. We then use this technique to establish the relationship to predict the class of income a person may belong to.

II. Naive Bayes Classifier

The goal in classification problem is to take an input feature $\boldsymbol{x}$ and assign it to one of the two classes (K classes in general). Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes Theorem with strong independence (naive) assumption between features. It is assumed that value of a particular feature is independent of any other feature.

By assuming Naive assumptions, we are essentially ignoring relationships among features. Because of such conditions, Naive Bayes is said to have *high bias* and this being one of the simplistic models, is said to have *low variance*.

Let $x_1, x_2, ..., x_D$ be the feature vector $\boldsymbol{x}$.

$$P(y = C_k|x_1, x_2, ..x_D) = \frac{P(y = C_k)P(x_1, x_2, ..x_D|y = C_k)}{\sum_{j=1}^{K} P(x_1, x_2, ..x_D|y = C_j)P(C_j)}$$

Assuming $x_1, x_2, ..., x_D$ are independent,

$$= \frac{P(y = C_k)\prod_{i=1}^{D} P(x_i|y = C_k)}{\sum_{j=1}^{K}\prod_{i=1}^{D} P(x_i|y = C_j)P(C_j)}$$

$$P(y = C_i|\boldsymbol{x}) \propto P(y = C_k)\prod_{i=1}^{D} P(x_i|y = C_k)$$

$P(y = C_k)$ is called the overall or the prior probability.

Now the challenge lies in identifying $P(x_i|y)$. Notice that because we assumed conditional independence, we now have to model only $P(x_i|y)$ individually for all features instead of modelling $P(x_1, ..., x_D)$ in the D-dimensional space. This greatly reduces computational complexity as we search only in 1 dimensional space d times instead of searching in a D-dimensional space one time (Volume expands cubically). This is illustrated mathematically in the below section.

$P(x_i|y)$ is generally assumed to be Gaussian for continuous data and Bernoulli distribution for binary data (Multinoulli/ Categorical distribution for categorical data). The accuracy could be improved by modelling $P(x_i|y)$ more accurately by using any density estimating methods.

A. Reasoning behind Naive Assumption

Let's for now do not consider conditional independence and assume the likelihood given a class k is from a Multivariate distribution.

$$X = (X_1, X_2, ..., X_D)$$

$$X|y \sim \mathcal{N}(\mu_k, \Sigma_k)$$

$$f(X|y=C_k) = \frac{1}{2\pi^{\frac{D}{2}}|\Sigma_k|^{0.5}} exp(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k))$$

where μ_k and Σ_k are class specific mean and covariance matrix. These mean and covariance matrix needs to be determined or in a machine learning linguistics, needs to be learnt from the dataset.

μ_k has D parameters and Σ_k has D^2 parameters that need to be determined. If there are K such classes, number of parameters that need to be estimated is

$$K(D+D^2)$$

This also means we need to have a huge dataset in order to estimate the parameters accurately (since MLE estimates are consistent and work well under large datasets). If we assume conditional independence, modelling $\boldsymbol{x}|y =$C$_k$ reduces to modelling $x_i|y = C_k$ for $i \forall D$

Assuming $x_i|y \sim \mathcal{N}(\mu_k, \sigma_k)$, only 2 parameters are unknown this time. For D such features and K such classes, the number of unknown parameters are just 2KD which is a lot lesser than earlier and has huge computational advantages.

It is to be noted that the latter method, though computationally advantageous, doesn't capture model complexity as the conditionally dependent one. This trade-off needs to be well noted in understanding Naive Bayes classifier. Hence Naive Bayes is a high bias but a low variance classifier.

B. Implementation

Let's understand naive bayes classifier for two classes (K=2). Recall,

$$P(y=C_1|X=\boldsymbol{x}) = \frac{P(y=C_1)\prod_{i=1}^{D}P(x_i|y=C_1)}{\sum_{i=1}^{2}P(y=C_i)\prod_{i=1}^{D}P(x_i|y=C_i)}$$

$$\text{And } P(Y=C_2|X=\boldsymbol{x}) = 1 - P(Y=C_1|X=\boldsymbol{x}) \quad (1)$$

Depending on the feature, we could assume various choices for $x_i|y$ distribution. Once we obtain $P(y=C_i|X=x)$, predictions could be done by

$$\hat{y} = \begin{cases} C_1 & \text{if } \hat{p} < Threshold \\ C_2 & \text{if } \hat{p} \geq Threshold \end{cases}$$

where C_1 and C_2 are the two classes. The threshold is usually set to 0.5.

C. Gaussian Naive Bayes

If the input feature is continuous and follows approximate Gaussian, we assume this distribution,

$$f_k(x|y=k) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)$$

where μ_k and σ_k^2 are the mean and variance for the k^{th} class (k=2 for our case).

$P(Y=C_1|X=x)$, $P(Y=C_2|X=x)$ is given by (1). We still do not know μ_1 and μ_2, σ_1 and σ_2 and have to be estimated.

We can use MAP estimates (Maximum A posteriori) estimates to estimate P(y) (Prior) and parameters. The estimates are given below:

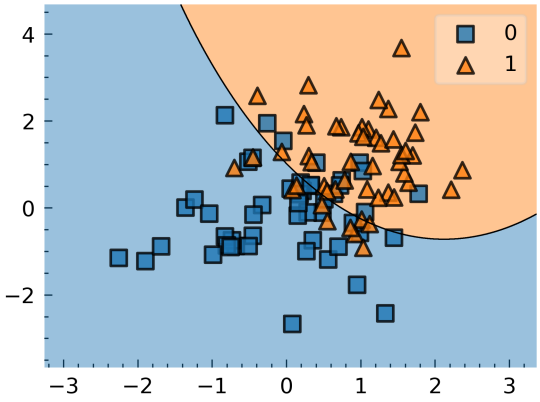
$$\hat{\mu}_k = \frac{1}{n_k}\sum_{i \forall n_k} x_i$$

$$\hat{\sigma}_k^2 = \frac{1}{n_k - 1}\sum_{i \forall n_k}(x_i - \hat{mu}_k)^2$$

$$Prior\ P(y=C_i) = \frac{n_k}{n}$$

n $\equiv$ total no. of observations

$n_k \equiv$ total no. of observations in the kth class



'The above plot represents the decision boundary of a Naive Bayes classifier obtained on a synthetic dataset of two features. The x_1 and x_2 used were generated from Gaussian distribution with a small correlation. Since the correlation was small and the distribution with a small correlation. Since the correlation was small and the distribution matches, naive bayes classifier does a good job of separating the classes. Note that unlike logistic regression, naive bayes don't produce a linear decision boundary.

The decision boundary would be linear if we had assumed a shared variance across classes instead of a different variance across each class.

D. Categorical Naive Bayes

When the input feature is categorically distributed, we use categorical Naive Bayes. CategoricalNB estimates a categorical distribution for each feature i of X conditioned on class y.

$$P(X_i|y=C_k) = \frac{N_{kt}}{N_k}$$

where N_{kt} is the number of times category t appears in the sample x_i which belongs to class k. N_c is the number of samples with class k.

Sometimes, when a new category hasn't been seen before arrives, the probability turns zero. In order to prevent this, a smoothing parameter, $\alpha > 0$ (usually 1) is added to the numerator and denominator.

$$P(X_i|y = C_k) = \frac{N_{kt} + \alpha}{N_k + \alpha}$$

E. Time Complexities

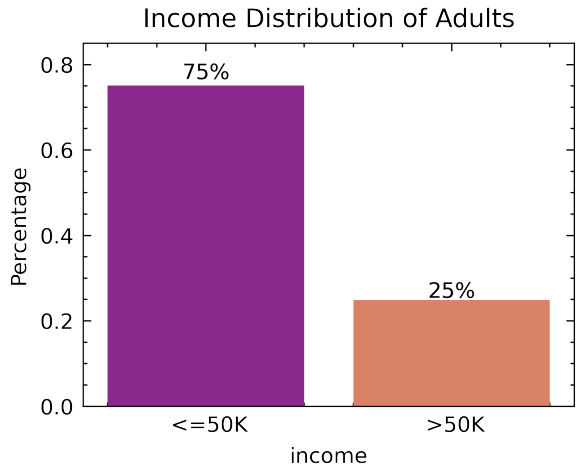
For a Categorical Naive Bayes training, the worst case is when we have n categories in n-datapoints (i.e. Each datapoint has a unique category), D dimensions and C classes. Then, inorder to compute and store all combinations, the required time complexity will be in the order of $O(n*d*c)$. The testing will require $O(d*c)$ as it involves just fitting and multiplying all d-dimensions for each class. Similar time complexity is obtained for Gaussian as well. Naive Bayes hence could learn fast and is computationally advantageous.

III. The Problem

In this section, We will analyse the Census bureau database and try to predict whether a person earns above $ 50 K per year or not.

A. Imbalanced Dataset

The dataset contains around 75% of people who earn below $50K and only 25% of people who earn above $50K



We know,

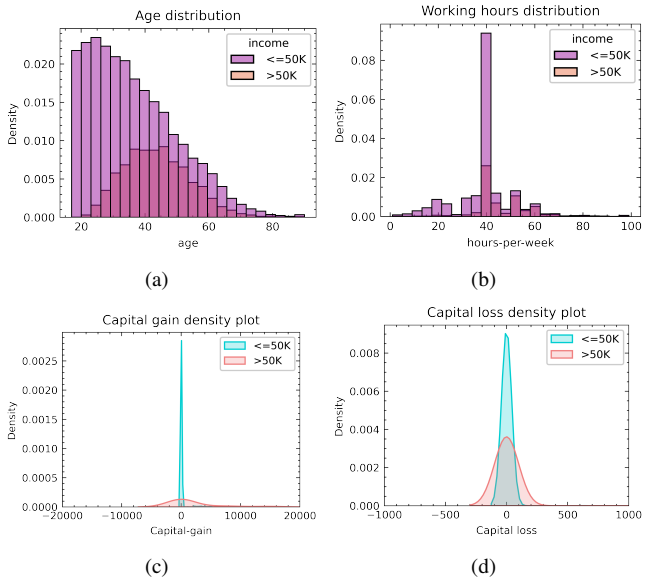
$$P(y = k|X = x) \propto P(y = k) \prod P(x_i|y = k)$$

Even though the likelihood probabilities are similar to some extent, the posterior probabilities are affected by the prior probabilities. We build two classifiers.

1) One with ignoring the prior probabilities (assuming equal weightage to both classes). We call it CLF1
2) Another with prior estimated through the dataset (0.751, 0.249). We call it CLF2.

Care is taken to ensure similar proportions are used in training and test set by doing a stratified split.

B. Numerical Attributes



It is important that inorder for Naive Bayes classifier to work well, the likelihood $X_i|y = C_i$ must be close to assumed Gaussian distribution.

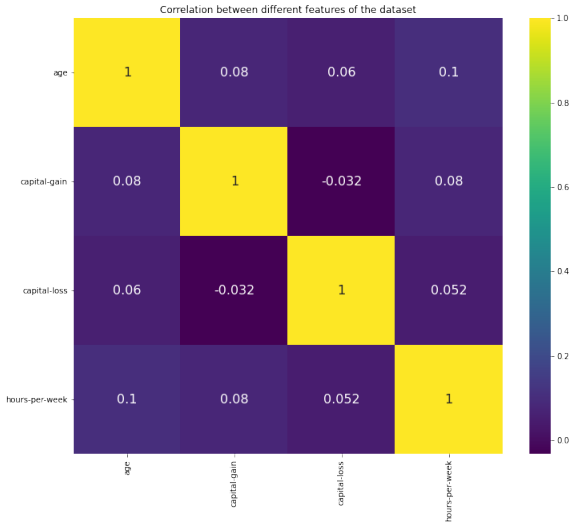
Age distribution, though shows some exponential feature, seems skewed and may not fit Gaussian perfectly. Also, more people earn above $50K at 45 years and above, when their career matures which makes sense. Box-cox transformation is a technique to make non-normal distribution into a normal-shape. Normality is an important assumption for many statistical techniques including Naive Bayes.

If $w_t$is our transformed variable and y is our target, then

$$w_t = \begin{cases} log(y_t) & \text{if } \lambda = 0 \\ \frac{y_t - 1}{\lambda} & \text{otherwise} \end{cases}$$

$\lambda$is varied from -x to x (-5 to 5) usually. The optimal value is the one which results in the best approximation of a normal distribution. Though the above formulation only works for y_t being positive, this could be easily fixed by considering $y_t + c$ instead of y_t Capital-gain and Capital-loss seem to follow a normal distribution. Hours-per-week seems a little bumpy but the normal distribution would be a decent approximation due to the spike in middle.

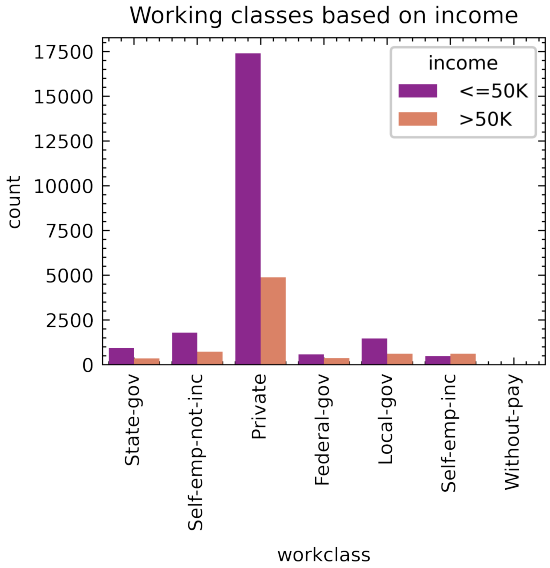
Education-num (Number of years of education) feature is dropped since we are including categorical education level feature and correlation could exist between them.



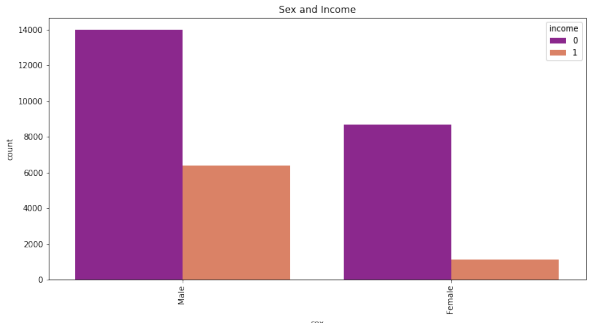
The above heat map reveals that the correlation between numerical features is very small. Since naive bayes assumes independence, in order for it to work well, this needs to hold good.

C. Categorical Features

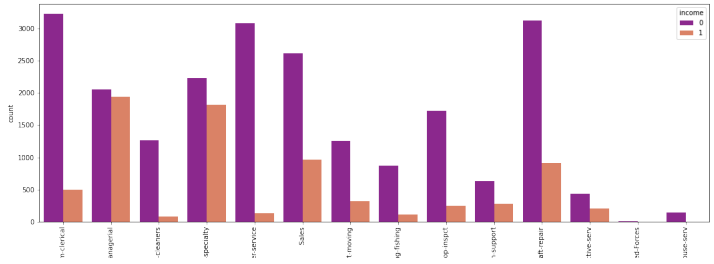
Insights regarding few categorical features are discussed below.



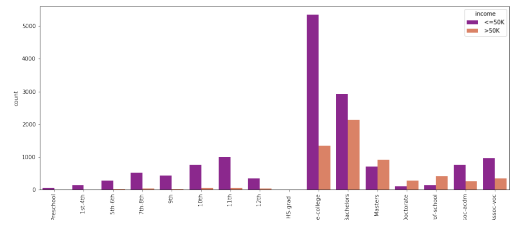
People in the private sector face significant differences in the pay scale. A lot of people earn below 50k and around one third earn above. There aren't as many differences that exist in the government sector compared to private ones. Self employed inc is the only category where there is more number of people earning above 50k than below. We could say when you handle your own business, start-up, you are more likely to earn better if it gets successful.



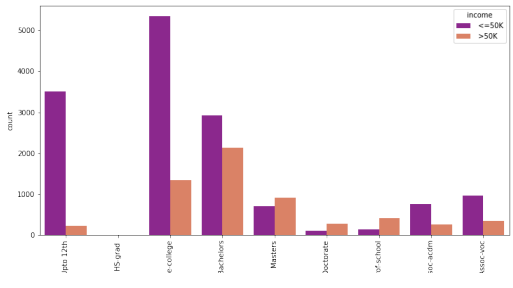
Less than 10 % of females earn more than 50K dollars a year while 33% of males earn more than 50K dollars a year.



Executive managerial role and Prof-specialty roles are likely to earn more than 50K dollars a year while clerical, Handlers-cleaners, Farming-fishing, Machine inspection, Transport-moving are less likely to earn a lot.



(e)



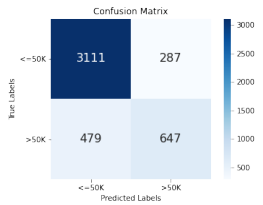
(f)

People who have completed less than 12th standard rarely earn above 50K. The distribution for people below 12th is

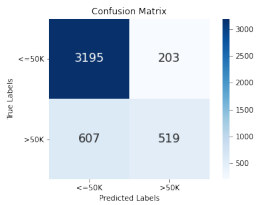
almost similar with most earning considerably less. We could group education less than 12th into a single group. This would make probability estimates accurate, especially useful in a naive bayes setting. People who have completed Bachelors, Masters, Doctorate, Prof-School significantly are on the higher pay scale implying education does matter to a certain extent.

D. Model Evaluation

After Data cleaning, Preprocessing and Feature Engineering, two Naive Bayes classifiers (one with uniform prior, another with prior learnt from data) were trained. The following confusion matrix was obtained on the test set.



(g) Uniform Prior- CLF1

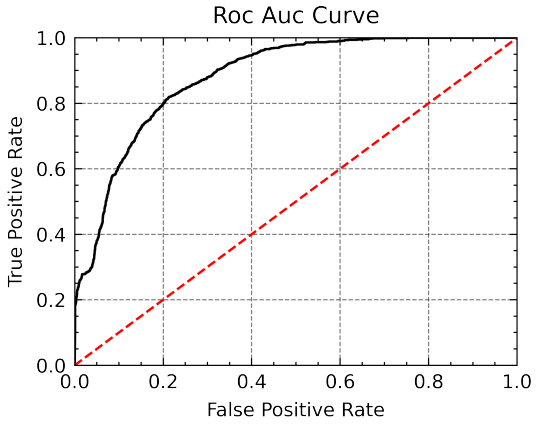


(h) Learnt Prior - CLF2

Naive Bayes Classifier		
Metric	CLF1 Score	CLF2 Score
Accuracy	0.8278	0.8177
Precision	0.6824	0.7107
Recall	0.5772	0.4516
F1 score	0.6254	0.5523

In a classification setting, accuracy may not be the best score to consider especially in a skewed class situation. The training data in our case is imbalanced. Precision is the ratio of correct positive predictions to the total positive predictions of our model. The recall is the ratio of positive instances that are correctly detected by our classifier. The F1 score is simply a harmonic average of these two. Classifier 1 (that assumes uniform prior for both classes) seems to perform better both in terms of Accuracy and F1 score than the classifier 2 (that learnt its prior)

ROC curve is another common tool used with binary classifiers. The ROC curve plots the true positive rate (another name for recall) against the false positive rate. The FPR is the ratio of negative instances that are incorrectly classified as positive. It is equal to one minus the true negative rate, which is the ratio of negative instances that are correctly classified as negative. The TNR is also called specificity. Hence the ROC curve plots sensitivity (recall) versus 1 – specificity.



The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner). One way to compare classifiers is to measure the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5. The area under the ROC curve of our Naive Bayes classifier is *0.8821*.

Both the classifiers have improved significantly after correcting for normality. The best classifier has improved approximately 0.7 percent in F1 score and also in Area under ROC. This reinstates how important distribution assumptions are in Naive Bayes setting.

E. Comparison with SVM

Let's try doing the same dataset with SVC's. This algorithm was chosen because it doesn't emphasis on Naive assumptions anymore and could capture non-linearities that we would get in Naive Bayes. A Support Vector Classifier with RBF kernel was gridsearched over γ and C.

Support Vector Classifier Comparison	
Metric	CLF Score
Accuracy	0.8566
Precision	0.6484
Recall	0.92
F1 score	0.7628

This clearly outperforms the Naive Bayes classifier at most levels. It is to be noted that the SVC classifier for single run in this dataset on the same machine took 57.6s on average while Naive Bayes was almost instantaneous. This is to conclude that even though SVM's performed better, Naive Bayes does possess discriminatory characteristics to separate out the classes and would be useful to set a minimum benchmark in large datasets.

IV. Conclusions

Based on our analysis, the key takeaways we had from this exercise are the following:

1) Naive Bayes assumes conditional independence and it may work well only when the assumptions are well met.

2) Imbalanced data could affect the predictions of Naive Bayes classifier. As in our case, we saw that adjusting to a uniform prior resulted in better predictions compared to the settings where no interventions were taken.
3) Factors such as age, education, working hours, sex and race seem to have impacted the income levels. While education and working hours are positives that could motivate an individual, issues related to disparities based on sex and race need to be addressed.

Possible avenues of research could include transforming numerical features to fit Gaussian well (such as box-cox), applying density estimation techniques for better fit and comparing results of other machine learning techniques to Naive Bayes classifier.

REFERENCES

[1] An Introduction to Statistical Learning, Gareth James et al. pp.138-151
[2] Aurelian Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, pp. 112–180
[3] Christopher M. Bishop, Pattern Recognition and Machine Learning

A Mathematical Essay on Decision Tree

Vignesh Kumar S
Department of Chemical Engineering
Indian Institute of Technology, Madras
Email address: ch18b118@smail.iitm.ac.in

***Abstract*—The purpose of this article is to understand the Decision Trees and the mathematics behind it. In this work, we demonstrate the application of the Decision Tree classifier illustrated through Car Evaluation Database. Due to the complex nature and the abstractness involved in classifying a car given its features, it is often difficult to obtain a good solution through manual approaches. Adopting a machine learning technique can help in obtaining insights and analysing different features in a large amount of data comprehensively. We try to establish a nexus between the nature of the car and different features involving its features.**

I. Introduction

In this paper, we will study Decision Trees, a technique predominantly based on tree-like models of decisions that only contains control statements. It is used to analyze and model either a dichotomous or multiple outcomes in classification setting and could also be extended as a regressor in regression setting. We will use this technique to classify the nature of a car using factors such as safety, capacity, costs etc.. This analysis would aid in obtaining insights among a large dataset and understanding the trend behind the nature of a car and various other factors.

The classification problem attempts to establish a relationship between a categorical target variable with one or more explanatory variable(s). Decision Trees are flow-chart like structure in which each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. Visually, Decision trees segments the predictor space into a number of simple regions using a rule based approach. In order to make a prediction for a given observation, we typically use the mean or mode of the training observations in the region to which it belongs. They are the fundamental units behind more powerful algorithms such as Random Forest and Adaptive boosting and form a crucial aspect in Machine learning.

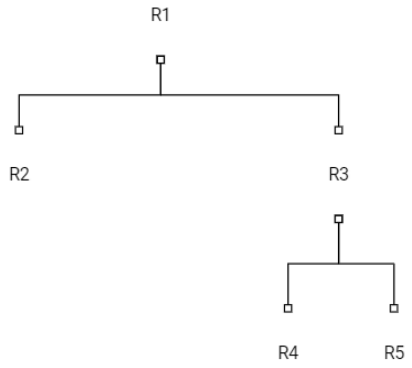
The dataset used for this problem comprises safety, luggage space, capacity, doors, maintenance and buying costs.. We use Decision Tree classifier to learn and predict whether the nature of the car (unaccountable, accountable, good, very good) given these input features.

This work represents the concepts behind Decision Trees and evaluation metrics involved. We then use this technique to establish the relationship to predict the class of of a car it may belong to.

II. Decision Tree

Decision Trees can be applied to both classification and regression problems. In this paper, we'll primarily look at it in a classification point of view but the following analysis could easily be extended into regression setting just by predicting the mean among the leaf nodes and using a squared loss function. Decision trees are easier to interpret and are often compared as a white-box model due to the ability we could compare its prediction. Later on, we'll see that combining many trees can result in powerful algorithms with improvements but with the penalty of loss in interpretation.

The goal in classification problem is to take an input feature $\boldsymbol{x}$ and assign it to one of the K classes. A Decision tree learns to segment the predictor space into simple regions with training dataset and when a new observation comes, it assesses where the new datapoint lies and make the predictions accordingly. Since there are no restrictions to the number of regions that could be made, a tree could grow long and have the potential to overfit every data point. Hence, Decision Trees are known to be low bias and high variance models.



R_2, R_4, R_5 are called Terminal nodes or leaves of the tree. The point along the tree where the prediction nodes are split is referred to as internal nodes.

A. Prediction via stratification of feature space

As we could see, decision trees are easier to visualize.

1) We divide the predictor space- that is the set of possible values for $X_1, X_2, ..X_p$ into J distinct and non overlapping regions $R_1, R_2, ..R_J$.
2) For every observation that falls into the region R_J, we make the same prediction, which is the most frequent observation for a classification problem while mean in regression problem in R_J.

Decision tree aims to find or learn the Regions $R_1, R_2, ..R_J$ that minimizes the cost function. Before considering the choices for cost function, it is important to understand that it is often computationally infeasible to find the best set of Regions or the global optimum. Decision Trees take a top-down, greedy approach with recursive binary splitting.

Generally for a classification setting, accuracy is often deemed as a standard metric that could be used as a loss function. However, for a decision tree, accuracy is not sufficiently sensitive and in practice two other measures are more preferred.

1) Gini Index/Impurity:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

This is a measure of total variance across the K classes. On close observation, this resembles to the variance of a Bernoulli distribution. This above can be equivalently written as:

$$G = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2$$

where $\hat{p}_{mk}$ is the proportion of training observation in the m^{th} region that are from the k^{th} class.

Notice that the gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one. So, small value implies that a node predominantly contains one class. And a zero value implies that the node is pure. Thus Gini index could also be viewed as a measure of impurity.

2) Entropy:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log\, \hat{p}_{mk}$$

Notice the close resemblance with the thermodynamics entropy formulation. Likewise, here too, entropy takes on a small value if the m^{th} node is pure or less random and large values if it is impure.

B. Learning

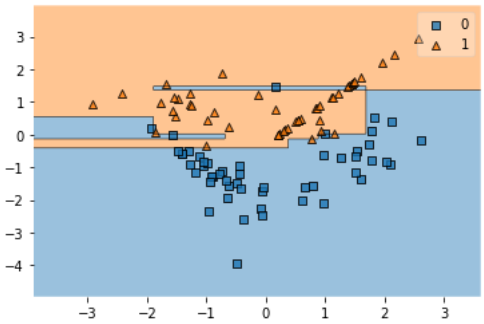
When building a Decision Tree, either Gini-index or entropy is typically used to evaluate the quality of a split. However, a split produces 2 nodes and in order to calculate the measure, a weighted average of leaf node is taken.

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

where m_{left} and m_{right} are the number of samples in left and right node and m is the sum of both. The split that results in the most reduction in the measure (Gini index/ Entropy) is chosen as the best split.

$J(k, t_k)$ could be viewed as the CART (Classification and Regression Trees) cost function. The algorithm first splits the training set into 2 subsets using a single feature k and a threshold t_k. A pair (t_k, k) is searched that minimizes the cost function the best. This procedure is repeated, looking for the best predictor and the best cutpoint in order to split the data further so as to minimise the cost function. This process continues till an appropriate stopping criterion is reached or when every training dataset is perfectly classified.

It is to be noted that without the intervention of appropriate stopping criterion or other means, the tree could grow deep, leading to overfitting. Decision Trees are also unstable, meaning trees learnt from different samples of a dataset are likely to differ in their structure compared to approaches such as Logistic Regression or Naive Bayes.



'The above plot represents the decision boundary of a Decision Tree classifier obtained on a synthetic dataset of two features. The decision tree learned is clearly overfit and doesn't seem to generalise well. In the coming sections, we'll look at ways to reduce the overfitting and improve the predictions.

C. Tree Pruning

A smaller tree with fewer splits might lead to lower variance and better interpretation at the cost of little bias. Let's consider a strategy to grow a very large tree and then prune it.

1) Cost Complexity Pruning:: Rather than considering every possible subtree, we consider a sequence of trees indexed by non-negative tuning parameter α. For each α, these corresponds to a subtree T $\subset T_0$ such that

$$J(k, t_k) + \alpha|T|$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of a tree. Notice that if α = 0, the subtree T is the original tree T_0. We can select a value for α using a validation set or cross-validation.

D. Bagging

Decision Trees suffer from high variance. Bagging is a procedure specifically useful for high capacity classifiers for reducing their variance. Bootstrapping is a technique to obtain datapoints using random sampling with replacement.

To apply bagging to trees, we simply construct B trees using B bootstrapped training sets and average the resulting predictions. The constructed trees however are grown deep and not pruned, hence suffer from high variance. But averaging reduces variance and results in a better model. This infact is a precursor to Random forest which differs as in addition to this, it deploys random features during splitting whose capability are not explored in this paper. Bagging improves prediction by lowering the variance. But the expense of loss of interpretability should be well noted.

E. Time Complexities

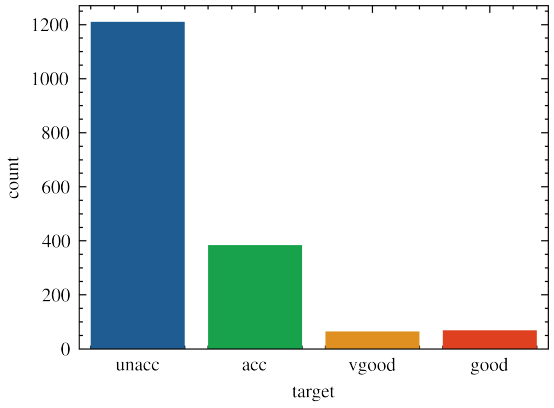
In a decision tree, a split has to be found until a maximum depth d_{epth} has been reached. In the worst case, maximum depth is equal to n, the training set size. The strategy for finding split is to look for each variable to different thresholds. Since there are p such variables, we make np decisions per node. Hence the upper bound for train time complexity is $O(n^2p)$

III. The Problem

In this section, We will analyse the Car Evaluation database and try to predict the nature of a car.

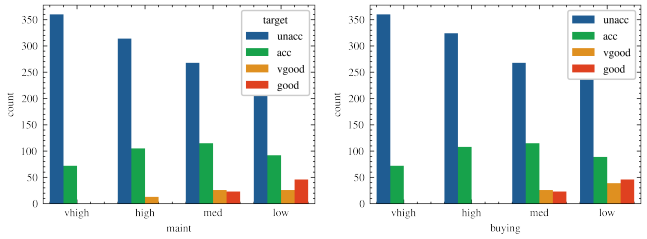
A. Imbalanced Dataset

The dataset predominantly contains class of type 'unacc'. Number of points containing classes 'vgood' and 'good' are considerably less. This is handled by giving appropriate weights to the decision tree learned from the data.



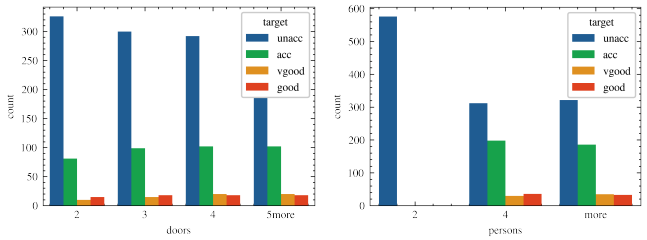
B. Attributes

From plot (a) and (b), We could infer that Most cars falls under unaccountable category. Cars that have high and vhigh buying and maintenance prices are generally unaccountable while the lower priced cars consists of all 4 categories. No good and vgood cars are present with vhigh buying or maintenance.



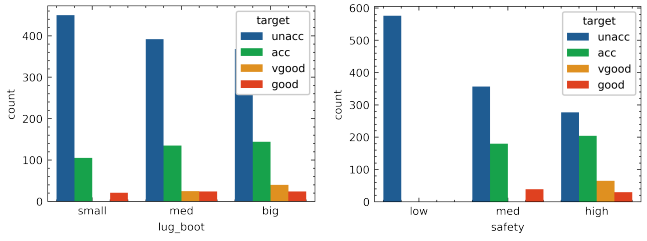
(a) (b)

From plot (c), Almost all categories have similar distributions. Number of doors doesn't seem to be a crucial criteria that could distinguish classes. From plot (d), Cars that have only 2 person capacity are unaccountable. 4 and 4 plus have similar distributions.



(c) (d)

From plot (e), Cars which have small lug boot are never 'vgood'. There has been mixed reviews across cars which have medium to big luggage. However, a trend of bigger space implying higher ratings are seen. From plot (f), Safety seems to play a major role. Every car which have a estimated low safety are unaccountable. No cars are classified as 'vgood' which have medium or low safety. A mixed reviews are seen across cars which were classified as high safety.



(e) (f)

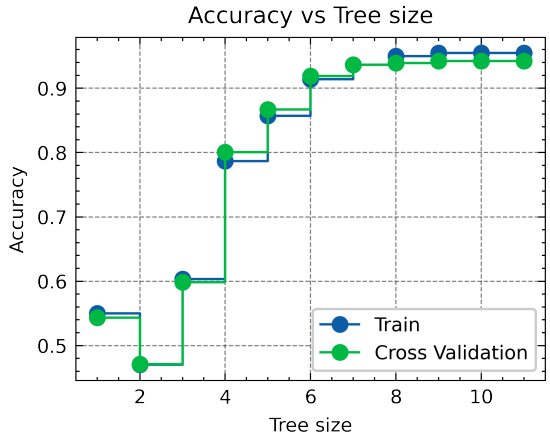
The dataset after checking for missing values is Label Encoded as the features found were categorical of type ordinal.

There exists some order in ordinal features which makes label encoding a better choice compared to one-hot encoding.

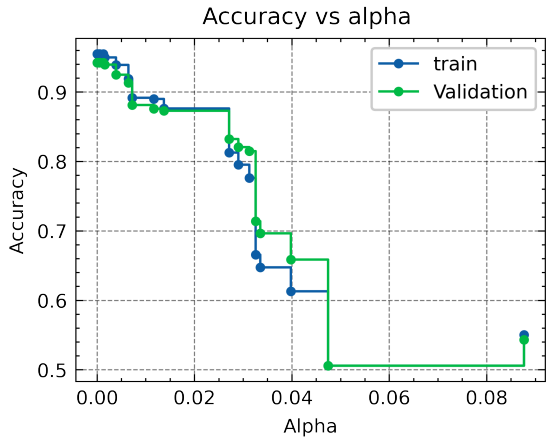
C. Model

The following models all assume a minimum number of samples required to split to be 20. This is kept so that model doesn't overfit too much and predictions could be obtained with confidence. However, this parameter could be experimented for different results.

1) CLF0: Decision Trees are constructed for varying depths. Significant improvements over accuracy is not seen beyond depth 7 with the cross validation set. We'll consider this classifier as the optimum clf0 with depth 7.



2) CLF1: Decision trees are trained with cost complexity pruning. A best α is chosen through validation set. The optimal α obtained was 0.00129.



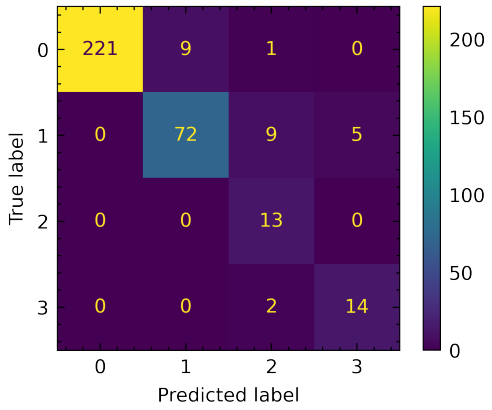
3) CLF2: The third classifier was constructed based on bagging 200 decision trees. Minimum samples required for a split was still considered to be 20 but there were no other restrictions used.

Decision Tree Classifier			
Metric	CLF0 Score	CLF1 Score	CLF2 Score
Accuracy	0.9364	0.9421	0.9104

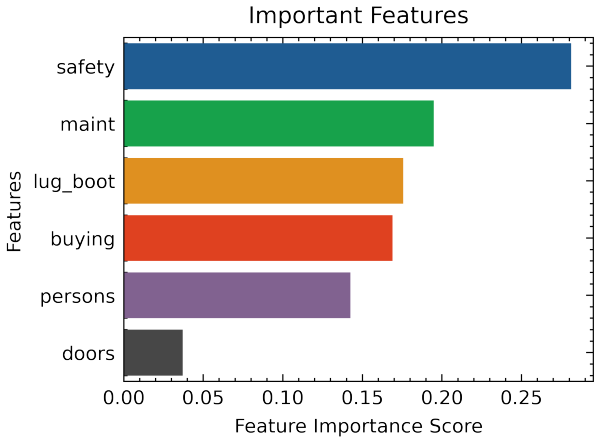
In a classification setting, accuracy may not be the best score to consider especially in a skewed class situation. The training data in our case is imbalanced. Precision is the ratio of correct positive predictions to the total positive predictions of our model. The recall is the ratio of positive instances that are correctly detected by our classifier. The F1 score is simply a harmonic average of these two. The precision, recall and f1 scores were also calculated for each classifier. However since they are of dimensions 4 (equal to the number of classes), they are not reported here. The classifier that performs the best comparing all the criteria in the cross validation set is the CLF1, the pruned classifier.

CLF1 is retrained, but this time on training and cross validation together and the model assessments on the train set are reported below.

4) Best Classifier: The best classifier CLF1 reports the following confusion matrix on the training dataset.



The classifier performs well on the test set. However, some difficulty is seen in getting the class label 2 and 3 accurately (class 'good' and 'vgood' respectively). This could be due to the fact that the dataset was skewed at the first place and hence the errors.



The above suggests that safety is the main feature followed by maintenance to predict the nature of the cars. The least important feature is found to be number of doors. Note that this

agrees with the intuition that we obtained through visualization of the dataset.

IV. Conclusions

Based on our analysis, the key takeaways we had from this exercise are the following:

1) Decision Trees are prone to overfitting. Without intervention, they tend not to generalise well and hence appropriate means should be employed to counter this.
2) Decision trees are easier to interpret. However, they are prone to instability and could vary with few differences in data. To address this, we discussed bagging, a method to reduce variance but at the expense of interpretability.
3) Safety is the most important feature that is considered for classifying the nature of a car. We also saw how just by visualizing, how a large amounts of insights were obtained.

Possible avenues of research could include trying out dimensionality reduction before applying decision trees, visualizing decision stumps and using boosting methods to improve the results.

References

[1] An Introduction to Statistical Learning, Gareth James et al. pp.303-323
[2] Aurelian Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, pp. 215–228
[3] Christopher M. Bishop, Pattern Recognition and Machine Learning

A Mathematical Essay on Random Forest

Vignesh Kumar S
Department of Chemical Engineering
Indian Institute of Technology, Madras
Email address: ch18b118@smail.iitm.ac.in

***Abstract*—The purpose of this article is to understand the Random Forest algorithm and the mathematics behind it. In this work, we demonstrate the application of the Random Forest classifier illustrated through Car Evaluation Database. Due to the complex nature and the abstractness involved in classifying a car given its features, it is often difficult to obtain a good solution through manual approaches. Adopting a machine learning technique can help in obtaining insights and analysing different features in a large amount of data comprehensively. We try to establish a nexus between the nature of the car and different features involving its specifications.**

I. Introduction

In this paper, we will study Random Forests, a technique predominantly based on collection of tree-like models of decisions that only contains control statements. It is used to analyze and model either a dichotomous or multiple outcomes in classification setting and could also be extended as a regressor in regression setting. We will use this technique to classify the nature of a car using factors such as safety, capacity, costs etc.. This analysis would aid in obtaining insights among a large dataset and understanding the trend behind the nature of a car and various other factors.

The classification problem attempts to establish a relationship between a categorical target variable with one or more explanatory variable(s). Random Forest consists of Decision Trees which are flow-chart like structure in which each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. In order to make a prediction for a given observation, we run the sample through the collection of Decision Trees, a.k.a the Forest and typically use the mean (in case of regression) or mode (in case of classification) of all the predictions to make the final prediction.

The dataset used for this problem comprises safety, luggage space, capacity, doors, maintenance and buying costs.. We use Random Forest classifier to learn and predict whether the nature of the car (unaccountable, accountable, good, very good) given these input features.

This work represents the concepts behind Random Forest and evaluation metrics involved. We then use this technique to establish the relationship to predict the class of of a car it may belong to.

II. Random Forest

In the earlier paper, we had discussed about Decision Trees. To recap, they can be applied to both classification and regression problems. In this paper, we'll also look at Random Forest in a classification point of view but the following analysis could easily be extended into regression setting just by predicting the mean among the leaf nodes and using a squared loss function. While decision trees are easier to interpret and are often compared as a white-box model due to the ability we could compare its prediction, by combining many such trees, we could improve the prediction accuracy of a tree with much lower variance but at the expense of some loss in interpretability.

The goal in classification problem is to take an input feature $\boldsymbol{x}$ and assign it to one of the K classes. Ensemble methods like Random Forests consists of predictions averaged over a number of Decision trees. When a new observation comes, it considers the majority vote or average value to determine the final prediction. Even though decision trees are known to overfit and hence a high variance model, averaging the predictions over many trees significantly lowers the variance and improves the nature of predictions. This is the principle behind Random Forest and it hence addresses the shortcomings of Decision Trees while exploiting its powerful nature of tree.

Decision Trees take a top-down, greedy approach with recursive binary splitting. It is important to understand that since Random Forest is a collection of many such trees, it is computationally infeasible to find the global optimum for Random Forest. Hence we take a greedy approach to minimize the cost function.

Generally for a classification setting, accuracy is often deemed as a standard metric that could be used while optimization. But we showed that for a decision tree, metrics such as Gini Index and Entropy are proven to be effective in practice. Random Forests being just a ensemble collection of parallelly trained trees, uses these metrics as well for determining the best split within each tree each step.

'The below plots represents the decision boundary of a Decision Tree and Random Forest classifier (made out of 100 decision trees) obtained on the same synthetic dataset of two features. The decision tree learned is clearly overfit and doesn't seem to generalise well. But the random forest seems to generalise better while not compromising the predictive capability. Thus random forests clearly seems to be a better version compared to a singly overfit decision tree.

Fig. 1: Decision Boundary produced by a decision tree

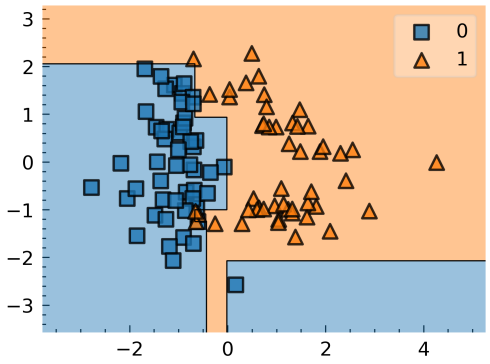
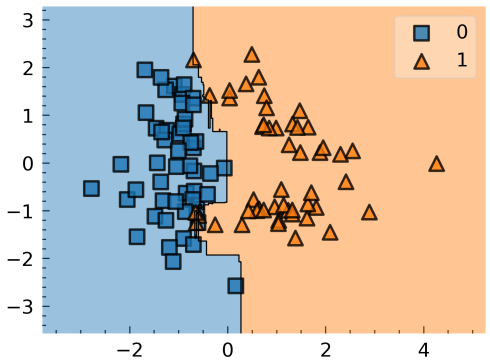


Fig. 2: Decision Boundary produced by a Random Forest



1) Metrics: Gini Index/Impurity:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$
$$= 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2$$

This is a measure of total variance across the K classes, where $\hat{p}_{mk}$ is the proportion of training observation in the m^{th} region that are from the k^{th} class.

Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one. A zero value implies that the node is pure.

2) Entropy:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log\, \hat{p}_{mk}$$

Notice the close resemblance with the thermodynamics entropy formulation. Likewise, here too, entropy takes on a small value if the m^{th} node is pure or less random and large values if it is impure.

When building a Decision Tree, either Gini-index or entropy is typically used to evaluate the quality of a split. However, a split produces 2 nodes and in order to calculate the measure, a weighted average of leaf node is taken. The split that results in the most reduction in the measure (Gini index/ Entropy) is chosen as the best split. This process continues till an appropriate stopping criterion is reached or when every training dataset is perfectly classified. For mathematical detail regarding to cost function of a Decision Tree, refer to the previous paper in the series. In this paper, we hence forth, will focus on bagging concepts and ensemble methods related to trees.

A. Bagging

Decision Trees suffer from high variance and can be very non-robust to dataset. A slight change or using a different sample from the dataset could lead to different trees. Bootstrap Aggregation or Bagging is a procedure specifically useful for high capacity classifiers for reducing their variance. Bootstrapping is a technique to obtain data-points. We generally do not have access to multiple training sets and hence we do bootstrapping, by taking repeated samples from the training data sets. Hence, we obtain distinct datasets by repeatedly re-sampling observations from the original dataset.This sampling is done with replacement, meaning the same observation can occur more than once in the bootstrap dataset.

To apply bagging to trees, we simply construct B trees $\hat{f}^1(x), \hat{f}^2(x), .., \hat{f}^B(x)$ using B bootstrapped training sets and average the resulting predictions to obtain a single low variance statistical model. The constructed trees however are grown deep and hence suffer from high variance. But averaging reduces variance and results in a better model. For classification, instead of averaging we take the most frequent predictions. Rest of the approach remains the same. B per se is not a critical parameter as B is just the number of trees we are averaging at. High values of B just implies we are averaging over many trees and hence apart from computational complexity, increasing B doesn't lead to overfit. In practice, B is kept sufficiently large so that the error has settled down.

B. Random Forests

Similar to bagging, we build a number of decision trees on bootstrapped training samples. But when building these Decision Trees, when each time a split is considered, a random sample of m predictors is chosen as split candidates from the full set of p-predictors.This also ensures a single feature is not influencing majority of decisions in all trees. Usually m $\approx \sqrt{p}$ but m is actually a hyper-parameter that could be chosen via cross validation. Prediction from the bagged trees are prone to be correlated. In order to prevent this from happening, we deploy such variation as above. This decorrelates the trees. The variance after averaging reduces in a much better fashion if the predictions from each tree are uncorrelated.

The main advantage of Bagging is that all the trees can be parallelly/ separately trained and hence could be sped up easily. But this kind of approach cannot be readily employed in sequentially/adaptively-trained models also known as boosting.

C. Out of Bag Error (OOB) Estimation

If we take n samples *with replacement* out of n samples, we can show that each bagged trees makes use of approx. $1 - \frac{1}{e}^{th}$ samples, or $approx. \frac{2}{3}^{rd}$ of observations. The remaining $1/3^{rd}$ of observations are left out in each trees. We can predict the response for the i^{th} observation using each of trees in which that observation was OOB or not present. This will yield around $B/3$ predictions for the i^{th} prediction. We can compute a single prediction easily by using an average or a majority vote. The resulting OOB error is a valid estimate of the test error for the bagged/ensemble Random Forest model.

D. Time Complexities

In a decision tree, a split has to be found until a maximum depth d_{epth} has been reached. In the worst case, maximum depth is equal to n, the training set size. The strategy for finding split is to look for each variable to different thresholds. Since there are p such variables, we make np decisions per node. Considering a Random Forest with n_{trees} such trees,

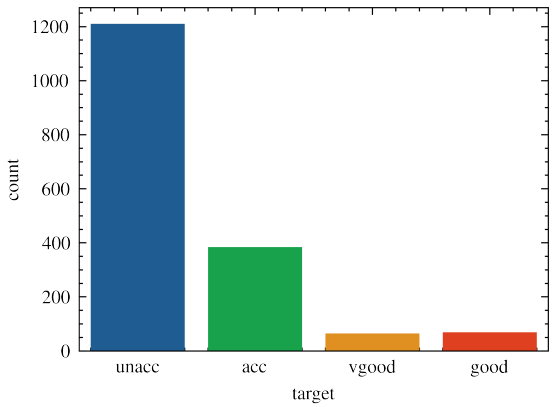
1) Assuming no-feature selection, the upper bound for train time complexity is $O(n_{trees}n^2p)$.
2) If $\sqrt{p}$ random feature selection is done at each node, the upper bound for train time complexity is $O(n_{trees}n^2\sqrt{p})$.

III. The Problem

In this section, We will analyse the Car Evaluation database and try to predict the nature of a car.

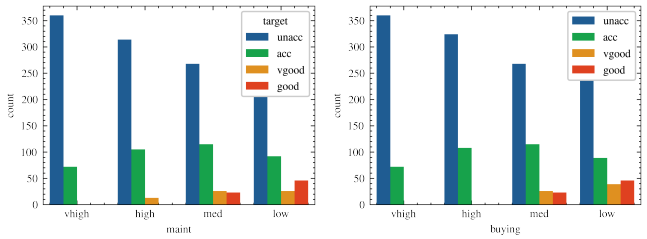
A. Imbalanced Dataset

The dataset predominantly contains class of type 'unacc'. Number of points containing classes 'vgood' and 'good' are considerably less. This is handled by giving appropriate weights to the Random Forest learned from the data.



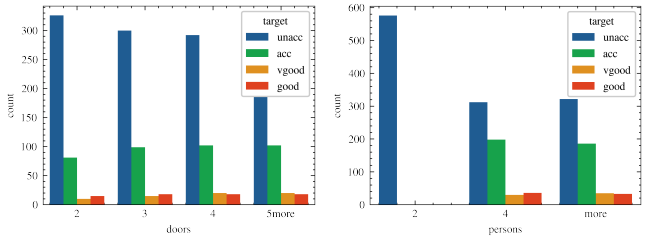
B. Attributes

From plot (a) and (b), We could infer that Most cars falls under unaccountable category. Cars that have high and vhigh buying and maintenance prices are generally unaccountable while the lower priced cars consists of all 4 categories. No good and vgood cars are present with vhigh buying or maintenance.



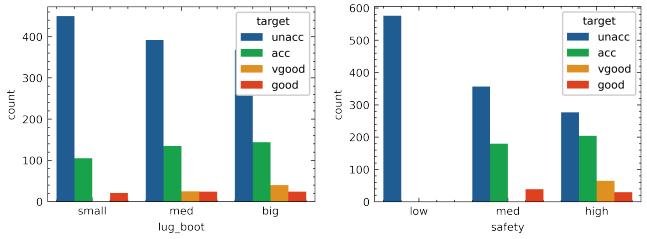
(a) (b)

From plot (c), Almost all categories have similar distributions. Number of doors doesn't seem to be a crucial criteria that could distinguish classes. From plot (d), Cars that have only 2 person capacity are unaccountable. 4 and 4 plus have similar distributions.



(c) (d)

From plot (e), Cars which have small lug boot are never 'vgood'. There has been mixed reviews across cars which have medium to big luggage. However, a trend of bigger space implying higher ratings are seen. From plot (f), Safety seems to play a major role. Every car which have a estimated low safety are unaccountable. No cars are classified as 'vgood' which have medium or low safety. A mixed reviews are seen across cars which were classified as high safety.



(e) (f)

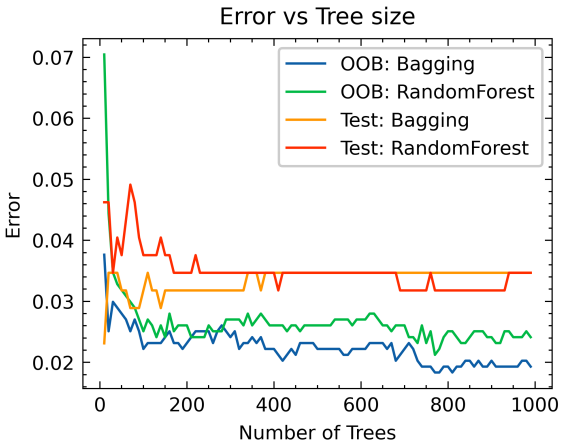
The dataset after checking for missing values is Label Encoded as the features found were categorical of type ordinal. There exists some order in ordinal features which makes label encoding a better choice compared to one-hot encoding.

C. OOB and Test Error of Bagged Model and Random Forest

Bagging and random forest results for the Car Evaluation Data is shown below. The test error (orange and red) is shown as a function of B, the number of bootstrapped training sets used. Random forests were applied with m = $\sqrt{p}$. The green and blue lines show the OOB error, which in this case is

considerably lower but looks like a decent approximation of the test error.

In this case, bagging (without variable selection or $m = p$) seems to perform tiny better than Random Forest ($m \approx \sqrt{p}$) itself. This may be due to the fact the trees are already uncorrelated and doesn't need random variable selection. We'll confirm this using grid-searching through different m's in a k-fold cross validation setting.



D. Model Evaluation

From the error vs tree size plot, 250 trees are sufficient enough where the errors stabilize pretty well. Searching over m (Number of features to consider when looking for best split) from 1 to 6 (total number of features) in 10-fold cross validation setting yields m as 6 (All features). This means for this particular dataset, Random forest without restricted number of features seems to perform marginally better over classifiers with feature selection. This strengthens the reasoning behind Error vs Tree size plot.

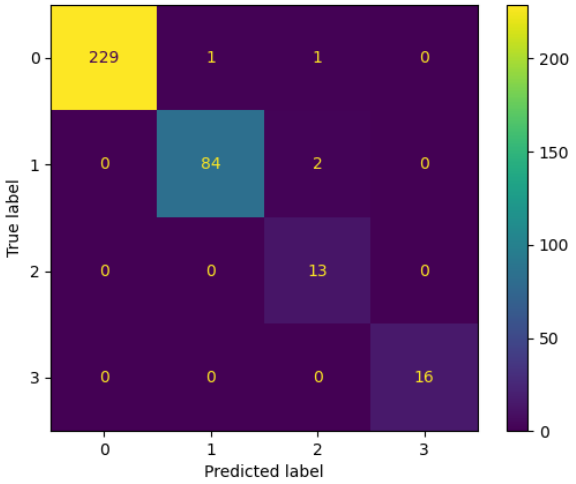
In a classification setting, accuracy may not be the best score to consider especially in a skewed class situation. The training data in our case is imbalanced. Precision is the ratio of correct positive predictions to the total positive predictions of our model. The recall is the ratio of positive instances that are correctly detected by our classifier. The F1 score is simply a harmonic average of these two.

The following metrics are reported for the best classifier on the test set.

Random Forest Classifier				
	Unacc	Acc	Good	Vgood
Precision	1.0	0.988	0.8125	1.0
Recall	0.991	0.976	1.0	1.0
F1-Score	0.995	0.982	0.8965	1.0

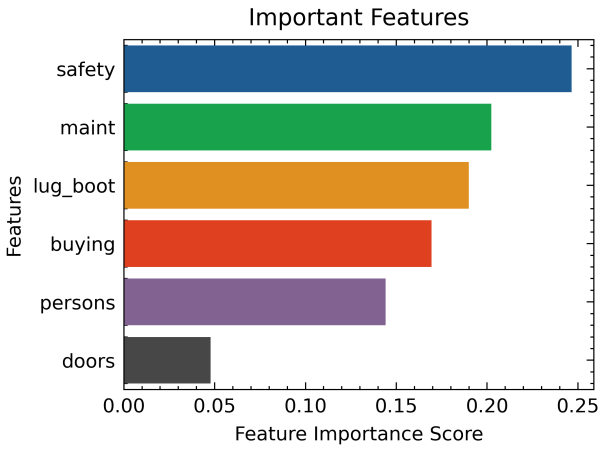
Accuracy = 0.988

The best random forest classifier (got through gridsearch) performs incredibly well over the current dataset. The F1-Score for class 'Good' may be low due to skewed data but apart from that, prediction accuracy seems too good.



The classifier performs extremely well on the test set. Naturally due to imbalanced dataset, the f1-score for class 2(good) is a little less than others but that's seems tolerable considering the skewness. However, the model performance seems too good to be true. This may be due to the fact that the target decision may indeed have arose due to control sequence like data generation. The true nature of the model can only be revealed when deployed in real time and assessing the results. Since the model is a ensemble and is bound to have lesser variance, we will go ahead with this final model unless we get further investigations.

Unlike decision trees, it is no longer possible to view and interpret it as a single tree like structure as this consists of two hundred fifty such trees and is an average over it. This is the trade-off between improved predictions over the expense of interpretability. However a summary of variable importance could be obtained using the Gini index that is decreased due to splits over a given predictor, averaged over all B trees.



The above suggests that safety is the main feature followed by maintenance to predict the nature of the cars. The least

important feature is found to be number of doors. Note that this agrees with the intuition that we obtained through visualization of the dataset.

IV. Conclusions

Based on our analysis, the key takeaways we had from this exercise are the following:

1) Random Forest are powerful learning algorithms that overcome the shortcomings of decision tree by using bagging and averaging techniques.
2) This comes over the expense of interpretability. However still, the model could be fairly interpreted but to a less extent.
3) Safety is the most important feature that is considered for classifying the nature of a car. We also saw how just by visualizing, how a large amounts of insights were obtained which were consistent with the result obtained through the algorithm.

Possible avenues of research could include trying out dimensionality reduction before applying Random Forest, visualizing decision stumps and using boosting methods to compare and contrast Random Forests.

References

[1] An Introduction to Statistical Learning, Gareth James et al. pp.303-323
[2] Aurelian Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, pp. 230–244
[3] Christopher M. Bishop, Pattern Recognition and Machine Learning

A Mathematical Essay on Support Vector Machine

Vignesh Kumar S
Department of Chemical Engineering
Indian Institute of Technology, Madras
Email address: ch18b118@smail.iitm.ac.in

***Abstract*—The purpose of this article is to understand the Support Vector Machine algorithm and the mathematics behind it. In this work, we demonstrate the application of the Support Vector classifier illustrated through Pulsar neutron star database. Due to the complex nature and the abstractness involved in identifying a pulsar star given its radio emission pattern, it is often difficult to obtain a good solution through manual approaches. Adopting a machine learning technique can help in obtaining insights and analysing different features in a large amount of data comprehensively. We try to establish a nexus to identify potential candidates for pulsar star given periodically obtained radio signal features.**

I. Introduction

In this paper, we will study Support Vector Machine, a technique used to analyze and model certain class, usually a dichotomous outcome. We will use this technique to automatically label pulsar candidates given statistics related to Integrated and DM-SNR profiles to facilitate rapid analysis. This analysis would aid in obtaining insights among a large dataset and understand the trend behind the possibility of identifying a neutron pulsar star.

The classification problem attempts to establish a relationship between a categorical target variable with one or more explanatory variable(s). Support Vector Machine follows discriminant modelling approach, which learns a function that maps each input to a class label directly from training data instead of modelling posterior probabilities like logistic regression or Naive Bayes classifier. This in general, is well suited for describing and testing hypotheses about features that could contain outliers. A Binary classifier is constructed by constructing a decision boundary and classifying inputs to classes based on the side it lies with respect to hyperplane.

The dataset used for this problem comprises mean, standard deviation, excess kurtosis and skewness of Integrated and DM-SNR profiles. We use Support Vector classifier to learn and predict pulsar candidates given these input features.

This work represents the concepts behind Support Vector Machine and evaluation metrics involved. We then use this technique to establish the relationship to predict the nature of a star it may belong to.

II. Support Vector Machine

The goal in classification problem is to take an input feature $\boldsymbol{x}$ and assign it to one of the two classes. The input space is thereby divided into decision regions whose boundaries are called decision surfaces. Support Vector Classifier without any extension is a linear classification model, which means the decision surfaces are linear functions of the input vector $\boldsymbol{x}$. However, with use of kernel tricks, a non-linear decision boundary could very well be obtained for enhanced use cases.

The support vector machine is a generalization of a simple and intuitive classifier called the maximal margin classifier, But this requires the dataset to be linearly separable which most datasets don't satisfy and cannot be applied. We then extend it to non-separable cases with this ideology and also introduce kernel tricks which enables a sophisticated way to model non-linear boundaries.

A. Maximal Margin Classifier

A hyperplane is geometry is defined as a subspace that is one dimension less than of its ambient space. Intuitively, one can think it as an extension of a line in a x-y plane or a plane in x-y-z dimensions to p dimensions.

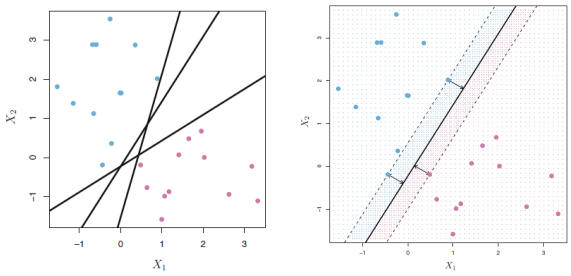
$$f(\boldsymbol{X}) = b + w_1 X_1 + .. + w_p X_p = 0$$

The above represents a p-dimensional hyperplane. Suppose we have n datapoints and p features, linearly separable with classes $y_i = \{+1, -1\}$, we want:

$$f(\boldsymbol{X}) > 0 \rightarrow R_1 \quad if\, y_i = +1$$
$$f(\boldsymbol{X}) < 0 \rightarrow R_2 \quad if\, y_i = -1$$

We can rewrite this into single equation as:

$$y_i f(\boldsymbol{X}) > 0$$
$$y_i(b + w_1 x_1 + .. + w_p x_p) > 0$$



(a) Infinite choices available to pick a decision boundary (b) The Maximal Margin classifier

A natural choice when infinite decision boundaries are possible is to pick the one which maximizes the margin, which is the perpendicular distance between the nearest datapoint.This

gives a generalization of a simple and intuitive classifier called the maximal margin classifier.

The given figure has 3 "support vectors". They *support* the maximal margin hyperplane in that sense that if these points are moved slightly, then the hyperplane would move as well. Moving other observations would not affect the hyperplane, provided that the observations does not cause it to cross the boundary set by the margin.

The original problem formulation could be written as:

$$\max_{w,b} \min_{i \in n} \frac{|w^T x_i + b|}{||w||}$$
$$s.t.\ sign(w^T x_i + b) = y_i$$

We could scale it such that the nearest point is at a unit distance from hyperplane and the maximization of $1/||w||$ could be re-written as minimization of $||w^2||$.

Hence,

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$s.t.\ y_i(w^T x_i + b) \geq 1$$

B. Soft-Margin Classifier

As stated earlier, the maximal margin classifier is not the way to go when the dataset is inseparable or even in the case of large datapoints, the margin would be very small and wouldn't allow any slackness. In order to obtain a greater robustness to individual observations, we aim to classify most of the training observations correctly and not every single one of them. This is the principle behind support vector classifiers a.k.a the softmargin classifier.

The margin is called soft now because it can be violated by some of the training observations. Rather than seeking the largest possible margin so that every observation is not only on the correct side of the hyperplane but also on the correct side of the margin, we instead allow some observations to be on the incorrect side of the margin or even on the incorrect side of the hyperplane.

The points that fall on the incorrect side of the hyperplane are misclassified. These points together with those that fall within and on the margin are the support vectors of the classifier and influence the decision boundary.

Let's introduce a variable ε_i, slackness, to capture this effect. If a point i falls outside the margin, ε_i = 0 and and $\varepsilon_i > 0$ for support vectors. The soft-margin problem could be written as:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \varepsilon_i$$
$$s.t.\ y_i(w^T x_i + b) \geq 1 - \varepsilon_i$$
$$\varepsilon_i \geq 0$$

This makes solution feasible for dataset which aren't linearly separable. Note that $\varepsilon_i > 1$ for misclassified point and $0 < \varepsilon_i \leq 1$ for points lying within the margin and on the correct side of the hyperplane and $\varepsilon_i = 0$ for the rest of the points.

C. Role of C

C determines the severity of the violations to the margin. If C$\rightarrow \infty$, the cost associated for misclassification is very high and there is no room for misclassification, hence results in hard-margin classifier. If C is very less, we allow more observations to violate the margin and also misclassifications.

In practice, C is treated as tuning parameter that is chosen via cross validation. When C is large, we seek narrow margins that are rarely violated. This amounts to a classifier that is highly fit to data, hence may have low bias and high variance. On the other hand, when C is small, the margin is wider, we allow more violations to it. This amounts ot fitting the data less hard and obtaining a classifier that is more potentially biased but lower variance. Hence C controls bias-variance trade-off.

D. Solution to optimization problem

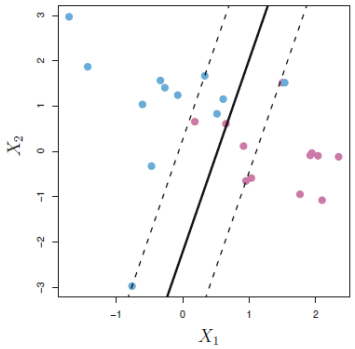
The Lagrangian of the above optimization problem in the primal space can be written as:

$$L(w,b,\varepsilon,\alpha,\beta) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \varepsilon_i + \sum_{i=1}^{n} \beta_i(-\varepsilon_i) + \sum_{i=1}^{n} \alpha_i(1 - \varepsilon_i - y_i(w^T x_i + b))$$

where α_i and β_i are lagrange multipliers and hence are ≥ 0. It seems that the above problem is easier to solve in dual space by applying minimax theorem whose details we won't delve into in this work. But the result of this optimization is interesting. The decision function obtained is:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$$

where $< x, x_i >$ is dot product and $\alpha_i \neq 0$ for support vectors only. Since the decision boundary is only dependent on support vectors, SVM is quite robust to behavior of observations that are far away (outliers) from the hyperplane unlike methods like LDA. (Linear Discriminant Analysis).



(c) Example of a soft margin classifier

E. The Kernel Trick

Also note that the decision function depends on dot product of support vectors only. We can rewrite the decision function as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

where K could represent the dot product. The intuition behind this is that we now would be able to transform X to a higher dimensional subspace and no longer need to remember that transformation but only interested in the dot product function of the resulting vector. Hence, We call this resulting function K as Kernel, which essentially is a transformation to a higher dimensional subspace but without needing to remember the transformation. This is advantageous computationally. The following are popular choices of kernel:

- $K(x_i, x_j) = x_i^T x_j$ Linear Kernel (no transformation)
- $K(x_i, x_j) = (1 + x_i^T x_j)^d$ Polynomial Kernel (Tranformation into polynomial space of degree d)
- $K(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2)$ RBF Kernel. (Transformation into Hilbert Space)

F. RBF Kernel

Given a test observation $x^* \equiv (x_1^*, .., x_p^*)^T$ is far from x_i, then $K(x_i, x^*) = exp(-\gamma ||x_i - x^*||^2)$ will be small.

Training observations that are far from x^* will essentially play no role in the predicted class label for x^*. This means that the radial basis kernel has very local behaviour. The farther the points from support vectors, the tiny the role it would play in f(x*). Hence, as γ is large, it makes very sharp turn (a.k.a) overfits.

G. More than 2 classes

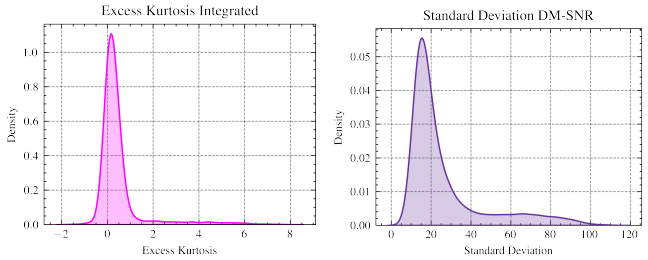
Unlike logistic regression, there is no natural extension to more than 2 classes that is possible. There are two roundabouts to deal with multiclass problem when we deal through SVM. The first is One versus One where for K classes, we construct $\binom{K}{2}$ classifiers for each pair. Then the final result is obtained through majority voting.

The second is by one versus all. We construct K SVM's where we label each class against (K-1) classes. We assign the observation to the class for which the signed distance from the hyperplane is largest as this amounts to a high level of confidence that the test observation belongs to the k^{th} class rather than to any of the other classes.

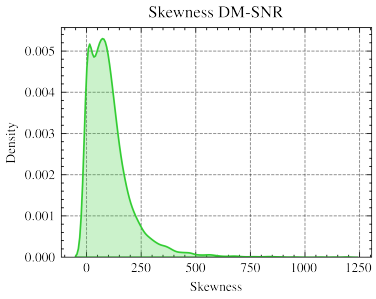
III. The Problem

In this section, We will analyse the Pulsar star dataset and try to predict candidates for pulsar star based on the information available. The support vector classification technique is used as a classification model in further analysis.

The dataset consists of statistics (Mean, standard deviation, excess kurtosis, skewness) of integrated and DM-SNR profile. Excess Kurtosis is a meaure of how fat a distribution's tail is when compared to the center of the distribution. Skewness is the degree of asymmetry observed in a probability distribution that deviates from the symmetrical normal distribution. Let's start by addressing any potential missing values.

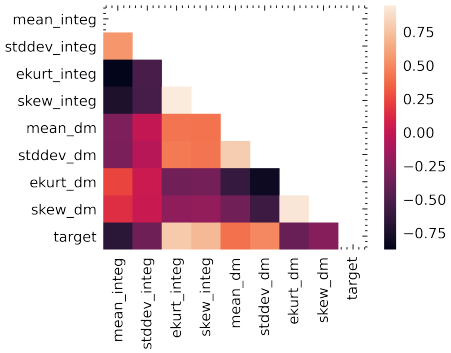


(d) (e)



(f)

The missing values are present in only the above three features and looking at these plots, the distributions of these features possess high tails, no clear patterns between mean, median/mode.

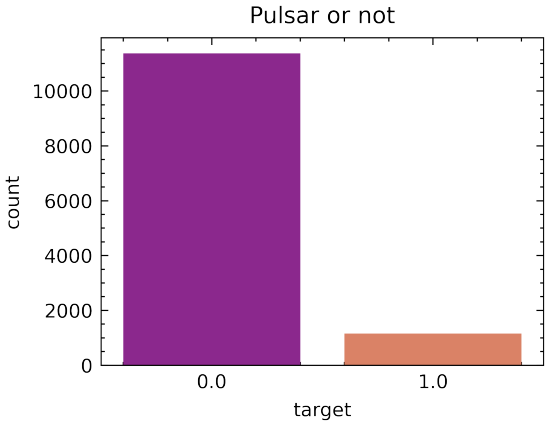


(g)

There is a fair amount of correlation between features and we could use this fact to impute the features instead of simple mean/median imputation. We hence use linear regression based imputer to fill the missing values. The descriptive statistics of both the imputed data and original data resembles (not shown due to space constraints) and hence this is a better strategy than constant imputation.

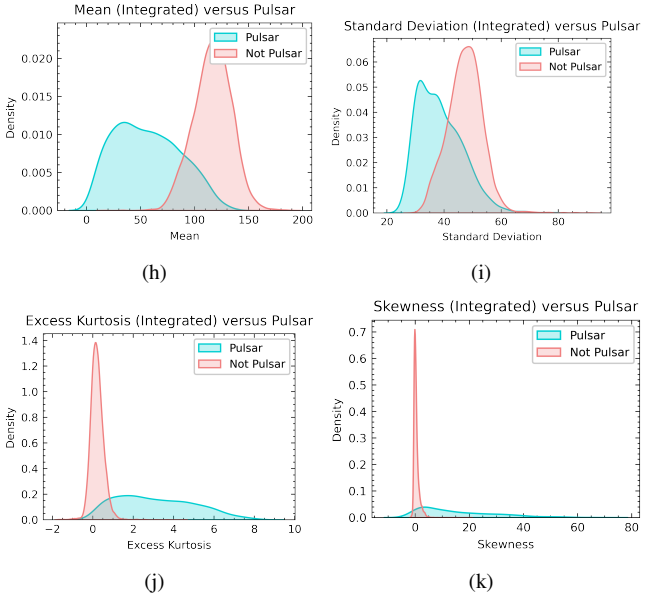
A. Exploratory Data Analysis

The given data is highly imbalanced. It only contains 9.2 % of pulsar stars and the remaining 90.8 % the non-pulsar ones.



This is taken care of by setting class weight to balanced while training.

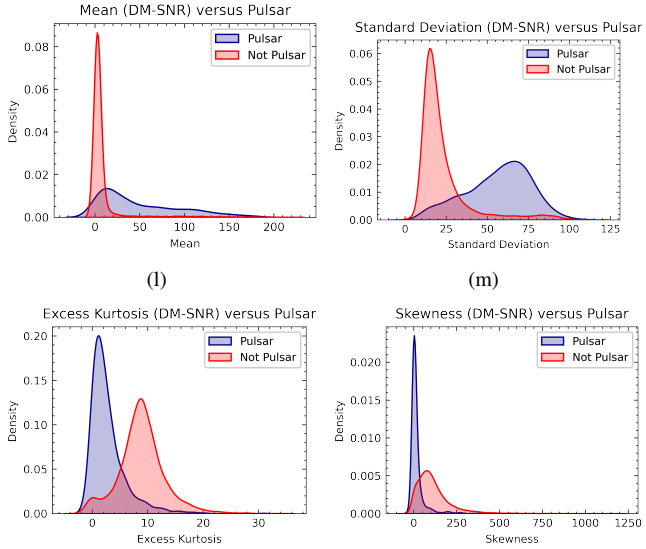
1) Integrated Profiles: We can generally say that mean and standard deviation of non pulsar inputs are high and excess kurtosis and skewness is generally lower in integrated profiles compared to pulsar stars.



(h) (i)

(j) (k)

2) DM-SNR Profiles: We can generally say that mean and standard deviation of non pulsar inputs are lower and excess kurtosis and skewness is generally higher in DM-SNR profiles compared to pulsar stars.



(l) (m)

(n) (o)

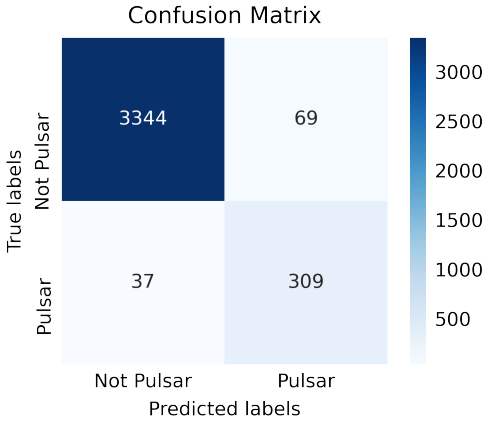
B. Data Preprocessing and Train test split

For SVM, data scaling is very vital as the distance from margin is sensitive to the scale of the feature used. We are going to do standardization here i.e, make the mean 0 and variance 1 of all the features. Also, since the dataset is imbalanced, stratified split is done to preserve the imbalancedness in train test split.

C. Model Evaluation

After data-preprocessing, a support vector machine classifier was grid searched between linear and rbf kernels on the training dataset with 5 folds of cross validation to maximise *f1-score*. This resulted in a best classifier with *rbf* kernel with C=3.703 and Gamma=0.0078. The classifier was trained on nearly 9000 training data points but had only a total of 15% (1488) support vectors implying only those many influenced the nature of decision boundary.

The following confusion matrix was obtained on the test/unseen dataset.



The classifier seems to perform good on identifying a pulsar dataset without many misses. We could see that it was unable

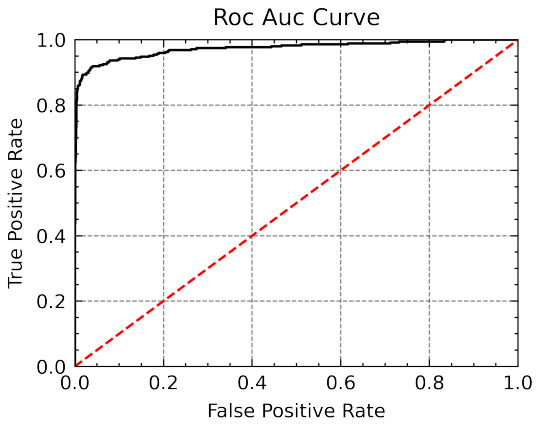
to detect only around 10% of them (37) though it incorrectly classified non-pulsar as pulsar for around 69 inputs. This could be tuned depending upon the knowing the real cost associated with misclassification but as of now this seems good enough.

In a classification setting, accuracy may not be the best score to consider especially in a skewed class situation. The training data in our case is imbalanced. Precision is the ratio of correct positive predictions to the total positive predictions of our model. The recall is the ratio of positive instances that are correctly detected by our classifier. The F1 score is simply a harmonic average of these two.

The following metrics are reported for the best classifier on the test set.

Support Vector Classifier	
Metric	Score
Accuracy	0.9718
Precision	0.8174
Recall	0.8930
F1 score	0.8535

ROC curve is another common tool used with binary classifiers. The ROC curve plots the true positive rate (another name for recall) against the false positive rate. The FPR is the ratio of negative instances that are incorrectly classified as positive. It is equal to one minus the true negative rate, which is the ratio of negative instances that are correctly classified as negative. The TNR is also called specificity. Hence the ROC curve plots sensitivity (recall) versus 1 – specificity.



The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner). One way to compare classifiers is to measure the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5. The area under the ROC curve of our Support Vector classifier is 0.9738.

Note that the standard ROC Curve requires varying the probability of score threshold of a classifier and obtaining the corresponding graph of the ordered pairs (TPR,FPR) for each varied threshold. But since SVM is defined in a such a way(Discriminant approach) that it doesn't produce probability, we can approximate the type of score by computing signed distance between the hyperplane. So the above is only an estimation and isn't the true nature of ROC curve. More accurate estimate could be obtained through Relevance Vector Machines which uses Bayesian methods to infer posterior probabilities of which isn't a particular interest in this specific work.

IV. Conclusions

Based on our analysis, the key takeaways we had from this exercise are the following:

1) Support Vector Machines are robust algorithms less sensitive to outliers and hence an enhancement of the classifiers we have studied so far.
2) Unlike most classifier algorithms, only a small percentage of dataset influence the decision boundary, called the support vectors. These improve the performance and lying farther away from margin can be yielded as a confidence metric.
3) Usage of Kernels is a clever way to search in higher dimensional space without increasing computational complexity and could potentially fit the data better.

References

[1] Aurelian Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, pp. 191–214
[2] Christopher M. Bishop, Pattern Recognition and Machine Learning
[3] An Introduction to Statistical Learning, Gareth James et al. pp.337–358