Time series analysis of financial data

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this final exam is to explore various avenues for generating value from a given set of data from financial markets. Due to the complex dynamics of financial markets, it is often not possible to predict how a stock will behave manually. The attempt here is to make use of machine learning models to make this prediction. All the mathematical machinery built so far in the course through assignments as well as some additional concepts not discussed so far will be used for the analysis. After gaining an understanding of the given data, we build a model for forecasting the prices. The accuracy of this model can help us understand how good our understanding is as well as help us predict future prices. Data visualization, cleaning and modelling is done using Python.**

***Index Terms*—time series, python, visualization, predictive modelling, financial data**

I. Introduction

In our everyday life, we often deal with a large amount of temporal data. Temporal data is characterised by the presence of an inherent ordering, which other forms of data lack. Often, the ordering itself is of great relevance and taking this into account is indispensable for any form of analysis whatsoever. This additional requirement often makes the analysis slightly more complicated. Though there are several techniques available for analyzing such forms of data, the use of machine learning to analyze temporal data has gained traction over the last few years, and we will be focusing on these methods.

Perhaps the most well-known example of temporal data is data from financial markets. Financial markets tend to be extremely complex with a lot of dynamics. Naturally, they generate a lot of data. Over the years people have tried out various methods for extracting value from this financial data. Deriving useful insights from financial data can be beneficial for the society as a whole. Financial data is often made available in the form of stock market prices and currency exchange rates.

The attempt here is to make use of machine learning techniques for analyzing data from financial markets. In machine learning parlance, temporal data is often referred to as time series data and the analysis of such data is hence termed time series analysis. We will be using some standard time series analysis techniques for analyzing the given stock market and currency data. Though no specific goal has been given, we will be trying to find the stock which can give the best return on investment, as this is the goal of any investor. We will also try to build a model to predict the price of stocks.

Section II gives an overview of the initial qualitative exploratory analysis. Often the amount of insights that can be gained through simple visualizations is underestimated. Section III gives a short description of the mathematical formalism behind time series analysis. Section IV describes the various time series analysis techniques and machine learning models models that were tried on the given dataset and the results that were obtained. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

The goal in this section is to do a qualitative analysis of the given stocks to determine the best choice for investment given only this data. Of course, actual investment decisions would require considering a multitude of other factors as well. However, as data scientists, our objective is to derive as much insights as we can from the given set of data.

We are given the price and volume data for six stocks: Cognizant, HCL, HDFC, ICICI, Infosys and SBI. We are also given the INR-USD exchange rates. Before moving on with the analysis, it is important to understand the features which have been provided to us:

- Opening price:This is the price of a stock at the time of opening of the market. This need not be identical to the previous day's closing price.
- High price: This is the highest selling price of a particular stock on that day.
- Low price: This is the lowest selling price of a particular stock on that day.
- Closing price: This is the price of a stock at the closing time of the market.
- Adjusted closing price: This is the closing price after adjusting for any actions which might have affected the price after the market closed.

There are two broad trading strategies: short-term trading and long-term trading. Short-term trading typically refers to strategies in which you hold only for short durations of time, typically days. Whereas in long-term trading, you are looking to hold the stocks for years. While in short-term trading it is necessary to consider all the different price information like opening price, high price etc since they are all indications of the volatility of the stock, in long-term trading you mostly deal only with any one price, typically the closing price, as you are considering the variation over a long time and the volatility within a day doesn't matter a lot. Since we are provided with data over the duration of years, we will be looking at long term performances. Hence, we will mostly be dealing with only closing prices.

A. *Trends and reversals*

We first plot the price evolution for each stock over the given period of time. Along with the closing price, we also plot the Simple Moving Average (SMA) of the price of the stock over the past 30 days. 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.

When the price of a stock is above the SMA line, we say that the stock is in an uptrend. This simply means that the stock is doing well as indicated by the rising prices. An uptrend usually validates a buying choice in the sense that if a stock that was bought is in an uptrend, the buying choice was right. Similarly, if a stock is below the SMA line, we say that the stock is in a downtrend. This usually indicates that the stock is performing poorly.

A change in trend is termed a reversal. Reversals often indicate the occurrence of strong economic activities. However, it must be noted that since SMA is based on past prices, the evolution of SMA is not an ideal metric for predicting future prices; rather it just confirms when a trend change has taken place.

We first take a look at the tech stocks. The charts for these stocks are given in Figures 1, 2 and 3.

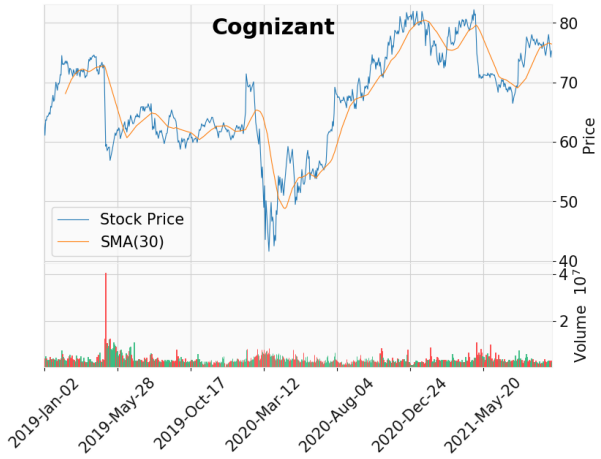

Fig. 1. Price chart for Cognizant

From a cursory look, it can be seen that the **HCL and Infosys tend to have a much more stable pattern of price evolution** when compared to Cognizant. They are also more consistently in an uptrend as opposed to Cognizant which is riddled with reversals. Also, we see two major price drops for Cognizant, one in May 2019 and one in March 2020. The one in March could be attributed to the economic slowdown, as we will see soon with the n=banking stocks. However, the drop in May seems to be something specific to Cognizant as a large fraction of shares were traded on the same day, indicating some company specific event. However, no further details can be understood without examining the economic situation at that time.

Fig. 2. Price chart for HCL

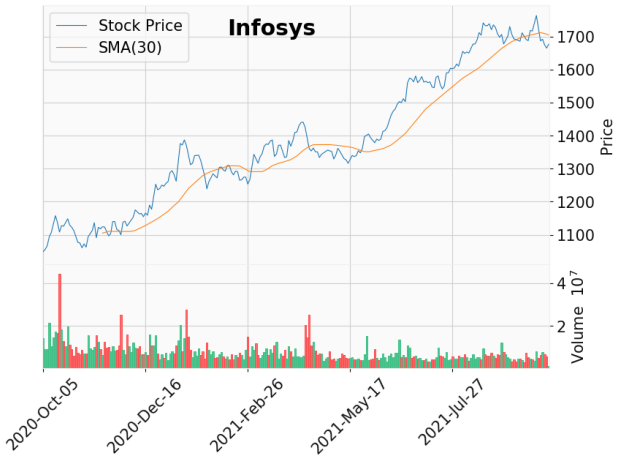

Fig. 3. Price chart for Infosys

We also look at the percentage increase in share prices over the whole of the given period. We observe that percentage increase in Cognizant share price over the period is **19.305%**, percentage increase in HCL share price over the period is **55.156%** and percentage increase in Infosys share price over the period is **59.888%**. Of the lot, Infosys and HCL seem to have done well as compared to Cognizant, which agrees with our intuition.

In any market, the time of entry (buying) and exiting (selling) are crucial. Although it practically impossible to predict the perfect entry and exit, looking at how much the ideal move would have made is often analyzed. To compute this, we look at the percentage change between the highest and lowest share prices for each stock. We obtain these to be **97.548%** for Cognizant, **68.814%** for HCL and **68.193%** for Infosys. This again shows that Cognizant is more volatile as compared to the other two stocks.

Next, we take a look at the banking stocks. The charts for these stocks are given in Figures 4, 5 and 6.

From a cursory look, it can be seen that all the three banking

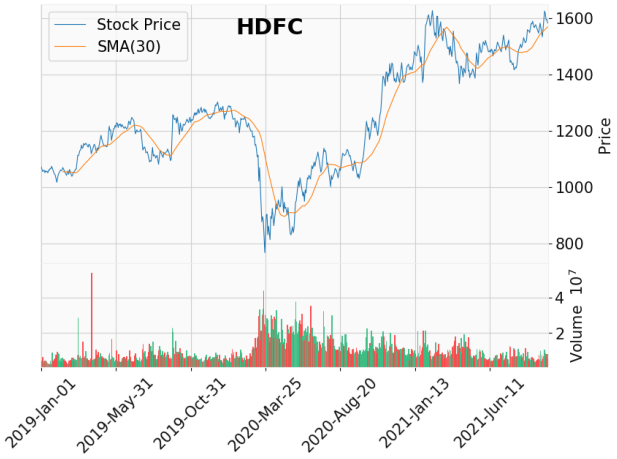

Fig. 4. Price chart for HDFC

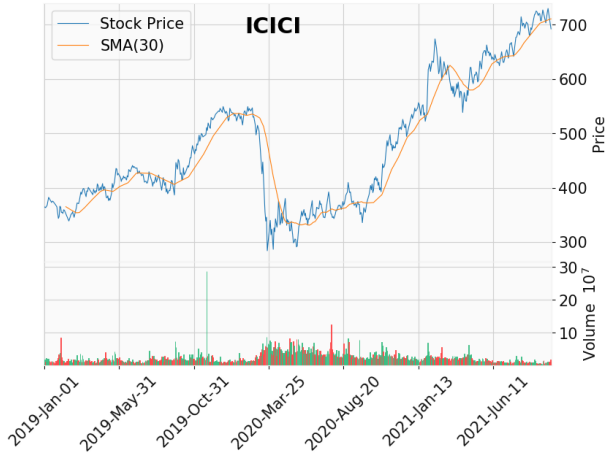

Fig. 5. Price chart for ICICI

stocks have roughly the same behaviour over the given period. All the three stocks tend to be riddled with reversals, with no trend being stable for a very long time. We also note that there is a huge dip in the prices of all the three stocks (as well as Cognizant as we saw earlier) around March 2020. This indicates that there was probably some crisis across the entire banking system. On investigation, it was discovered that this was probably due to the 'synchronised slowdown' in the World Economy. However, the stocks did recover after this slowdown and there hasn't been such a huge dip at any point after that.

We also look at the percentage increase in share prices over the whole of the given period. We observe that percentage increase in HDC share price over the period is **47.358%**, percentage increase in ICICI share price over the period is **90.336%** and percentage increase in SBI share price over the period is **50.750%**. Of the lot, the highest increase is seen for ICICI, for which the price almost doubled after the duration under consideration.

In any market, the time of entry (buying) and exiting

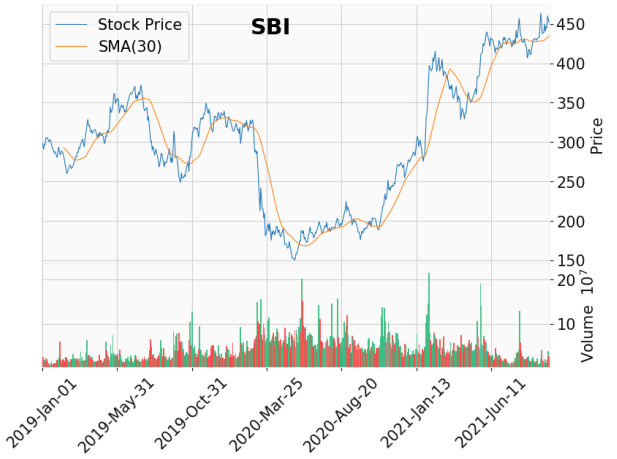

Fig. 6. Price chart for SBI

(selling) are crucial. Although it practically impossible to predict the perfect entry and exit, looking at how much the ideal move would have made is often analyzed. To compute this, we look at the percentage change between the highest and lowest share prices for each stock. We obtain these to be **111.886%** for HDFC, **157.042%** for ICICI and **207.391%** for SBI. We see that the values are all much higher than the tech stocks. This is probably due to the dip in March 2020, as someone who bought stocks then would have made a huge profit.

The charts also demonstrate the importance of having a diversified portfolio. Though the banking stocks seem to give very good returns, all of them crashed around March 2020, whereas tech stocks mostly did not. Having a diverse portfolio would help the investors stay afloat in such scenarios.

For the sake of completion, we also look at the currency exchange rates. The chart is shown in Figure 7. As we saw earlier, we see a huge decline in the value of INR in March 2020. This could again be ascribed to the economic slowdown. We also see that over the years, the value of INR has been decreasing.

B. Price and market capitalization

To get a comprehensive idea regarding the performance of all six stocks, it might be a good idea to chart the price evolution of all the stocks on the same chart. Note that the period for which information is available on the prices is somewhat different for different stocks; hence we plot for only the common time duration. The plot is shown in Figure 8.

Most stocks except Cognizant seem to have significant uptrend, at least in the first glance. HCL and Infosys seem to have a steady growth. HDFC seems to have a good uptrend but shows little stagnation in recent times. Though the are good indicators of how the companies perform, we should also consider Volume to get an estimate of amount obtained through shares. For this, we look at the market capitalization, which is the product of price and volume. The plot for market capitalization is shown in Figure 9.

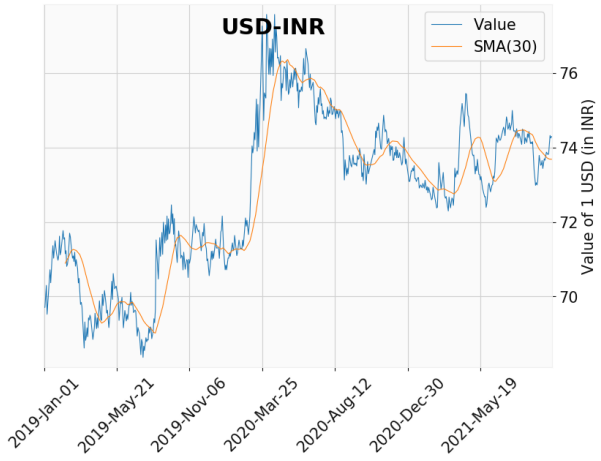

Fig. 7. USD-INR exchange rates

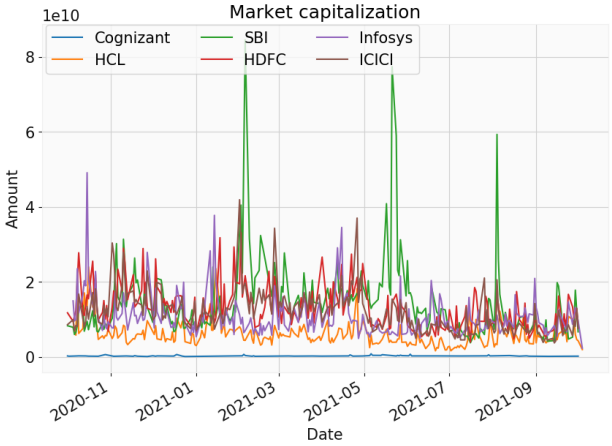

Fig. 9. Market capitalization across stocks

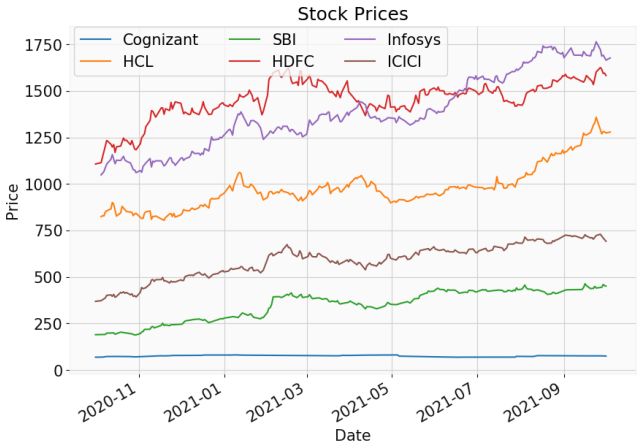

Fig. 8. Prices across stocks

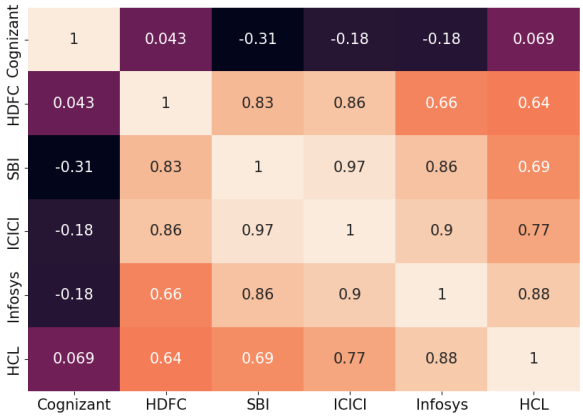

Fig. 10. Correlation across stock closing prices

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

We also display the correlation between the closing prices of stocks in Figure 10. Typically, we expect one stock to go up as another goes up from the same sector. This can be seen very evidently in the case of banking stocks, as the three of them have very high correlations. The trend is also followed to some extent by HCL and Infosys. Cognizant has very different behaviour, showing little to no correlation with any other stock whatsoever. This could be due to the unstable nature of the stock.

C. Rate of return

The main metric which is of importance to any investor is the Rate of Return (ROR). One would like to earn significant profits per rupee invested, taking into account the fact that for most individuals investment is limited by their financial potential. If $v[k]$ is the cost of stock on the 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 comparatively risk-free options such as Fixed Deposits (which have returns almost equal to the inflation rates) are potential options, it is useful to analyse rate of return by comparing with the inflation rate, which can be found through the changing value of INR. If $r[k]$ is the value of INR n the k^{th} day,

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

The charts shown in Figures 11, 12, 13, 14, 15 and 16 display the Return of Investments per rupee invested, accounting for change in INR value. The plots seem to closely follow normal distribution centred quite close to zero. The width of the distribution is representative of the volatility of the stock.

TABLE I
RATE OF RETURN

Stock	Mean Returns (10^{-3})	Standard Deviation
Cognizant	0.476	0.021
HCL	1.836	0.017
Infosys	1.906	0.014
HDFC	0.615	0.020
ICICI	1.102	0.026
SBI	0.995	0.025

Assuming we are looking for a low risk stock, Infosys seems to be a good option. Table I mentions the mean returns per day per rupee invested and the associated risk (standard deviation) of the stock analysed in an year-long period.

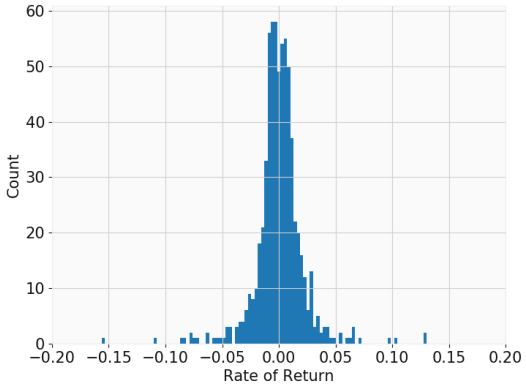

Fig. 11. Rate of returns for Cognizant

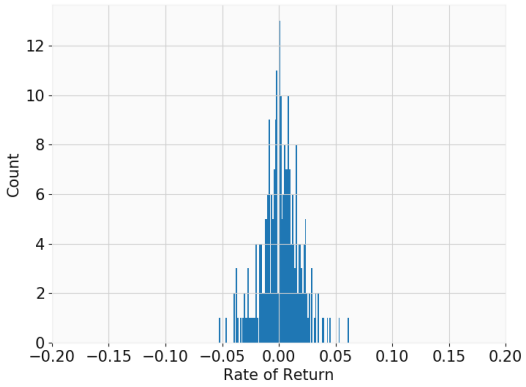

Fig. 12. Rate of returns for HCL

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 planning to hold the stock for a relatively long period of time.

Assuming that our goal is to choose a stock with maximum gains for minimal risk, we will go ahead and choose Infosys and try to forecast the prices using time series analysis. Note

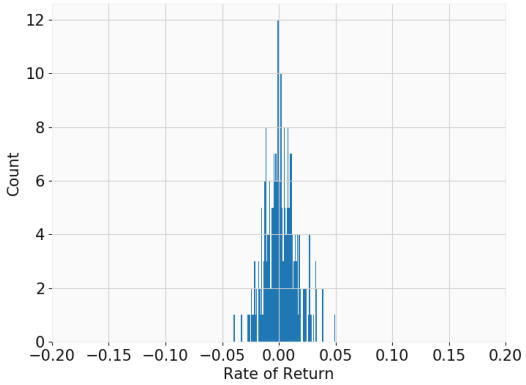

Fig. 13. Rate of returns for Infosys

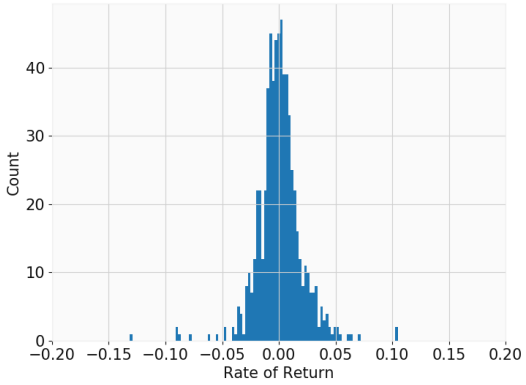

Fig. 14. Rate of returns for HDFC

that the ideal selection of stock would vary depending on variety of factors including risk tolerance of a person, how stocks in a portfolio reacts to the same market situation, corporate announcements and many more 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.

III. MODEL: ARIMA AND LSTM

A. Time series analysis

A time series is a sequence of data points that occur in successive order over some period of time. Time series tend to have a natural temporal ordering, and hence this can be contrasted with cross-sectional data which captures data from 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 past performance.

A set of datapoints $v[1], v[2], ..., v[k-1]$ is likely to be a time series if $v[k]$ depends on the past data. We must note that this dependence on past data is not necessarily long-term,

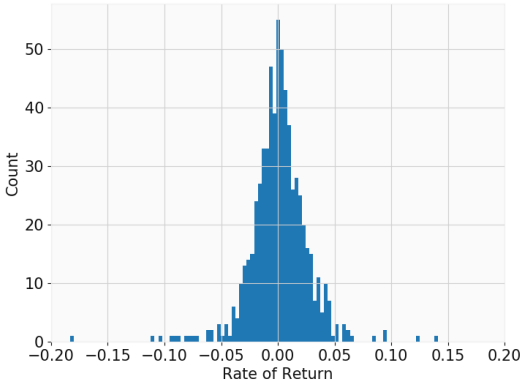

Fig. 15. Rate of returns for ICICI

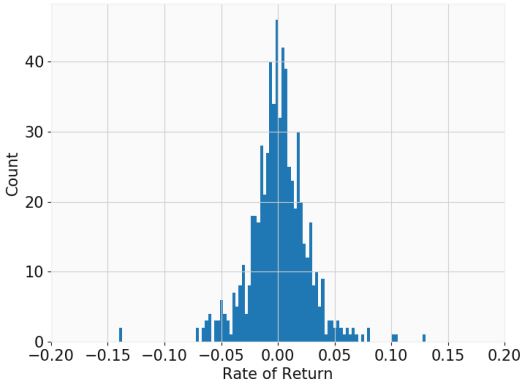

Fig. 16. Rate of returns for SBI

i.e., it is not necessary that $v[k]$ depends on values from the distant past like $v[1]$ for instance. Sometimes data could also depend on current and past residuals. The effect of past data could also be dying, in other words, could get weaker and weaker as we go more and more into the past. An example of this would be weather, where the today's weather is strongly dependent on yesterday's weather but weakly on the weather before two months.

Stationary processes: A process is said to be stationary if the statistical properties of the process are invariant with time. More formally, the process must satisfy:

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

where f is probability density function for the process. Often it is difficult to find processes that are strictly stationary in practice, so a weaker definition of stationary processes could be taken as invariance up to second order moments. This means that the mean of the process as well as the variance should be independent of time.

At this point, it is necessary to introduce a quantity termed autocovariance. Autocovariance is nothing but the covariance matrix of the data itself, computed by considering each sample instance separately. For example, consider two instances $v[k]$ and $v[k+l]$. The autocovariance $\sigma(k, k+l)$ for this pair of samples is $var(v[k], v[k+l])$. Technically, in order to calculate auto-covariance for the entire signal, we would need the distribution of each $v[k]$. However, if we assume that the process is stationary in a weak sense, we can consider $\sigma(k, k+l)$ to be independent of sampling time k and hence the quantity can be computed irrespective of k by considering only the time lag l between the pair. It should also be noted that if the correlation is computed instead of variance, the quantity is then termed autocorrelation.

Now, any signal can be decomposed as $v[k] + e[k]$, where $v[k]$ is the predictable/signal component and $e[k]$ is the unpredictable/noise component. In an ideal model, it is common to assume that $e[k]$ is uncorrelated to other sampling instants. This would mean that we expect the ACF of the errors (Auto Correlation Function) to be 1 only at a lag of 0:

$$\rho_e[l] = \begin{cases} 1 & \text{if } l = 0 \\ 0 & \text{otherwise} \end{cases}$$

MA process: When the current prediction depends on the noise/unpredictable component of current instance and of past instances, the process is said to be an MA process(Moving Average process), We assume that the process is weak stationary. An MA process of order 1 depends only on the noise component of the previous instance, while an MA process of order M depends on M of the previous instances. Mathematically, we have:

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

We have already seen that the ACF of $e[k]$, the noise component, is 1 only at lag 0 and 0 otherwise. Here, since $v[k]$ depends on past M noise components, the ACF of v[k] zeroes after M instances,i.e.,

$$\rho[l] = 0 \;\; \forall \;\; l > M$$

This is termed abrupt zeroing of ACF after M instances.

AR process: When the present data depends purely on the past instances, and the process is weak stationary, then the process is said to be an AR process (Auto Regressive process). An AR process of order 1 depends only on its immediate past, whereas an AR process of order p depends on the previous p instances, i.e., we have:

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

where $e[k]$ is the noise/unpredictable component. Unlike MA process, the ACF for an AR process does not zero 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 for AR processes, we take the conditional correlation so that link between $v[k]$ and $v[k-l]$ is broken. This is called Partial Auto

Correlation Function (PACF). For an AR process, PACF goes to zero after p instances and the ACF decays exponentially.

The important aspect of ACF and PACF graphs is that they help up in understanding the type of process and the mathematical model behind it even though it may not be known to us beforehand. By analysing the nature of ACF and PACF plots, it is possible to come up with a linear time series model.

ARMA process: In practice, a time series process could be a combination of both MA and AR processes, i.e., current predictions could depend on both past data and past residuals. Such a process is called an ARMA process. This can be represented mathematically as:

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

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

It is often the case that the assumption of a process being weak stationary does not hold condition. For many processes, the mean and variance could change with time. An example of such a process would be $v[k] = v[k-1] + e[k] + c$, where $e[k]$ is noise. This process tends to accumulate with time and grow upwards. Hence, it cannot be classified as an ARMA process. However, the difference $v[k] - v[k-1]$ is clearly an MA/AR process of order 0. Here, on taking the difference once, the process reduces to a stationary process and hence the process is said to be integrating of order 1. If D differences are needed to reduce a process to a stationary process, the process is said to be integrating of order D.

ARIMA process: Let us represent one-time differencing $v[k] - v[k-1]$ as $\nabla v[k]$. Then if

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

where ∇^D denotes D-time differencing and $w[k]$ is an ARMA process of order (P, M), then we say that the $v[k]$ is in an 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).

So far we have discussed time series analysis quite extensively, however we still haven't answered how all this analysis connects to the machine learning techniques we covered as part of this course. The idea is that once a time series model is determined, the problem reduces to a simple linear regression problem of estimating the coefficients. For example, let's say a given process is an AR process of order 2, i.e., $v[k] = d_1 v[k-1] + d_2 v[k-2] + e[k]$. Once this is known (by making use of visualizations of 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 techniques of linear regression.

B. LSTM

LSTM, which is an acronym for Long Short Term Memory, is one of the most recent algorithms which can be used for time series analysis. LSTMs essentially belong to the class of neural networks which rose to prominence over the past decade. Over the past few years, neural networks have become ubiquitous, finding applications in a huge number of domains. Depending on the domain of application, several structures of neural networks have evolved over the years, including fully-connected networks, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Of these, RNNs are of relevance to us as they evolved to deal with sequential data. Since any time series is essentially sequential data, RNNs can naturally be extended to time series data. However, traditional RNNs suffer from a problem known as the 'vanishing gradients' problem. What this means is that as the network grows in size, which is typically required by long-term time series data, the gradients start diminishing significantly which brings down model performance significantly.

LSTM networks were designed specifically to overcome this long-term dependency problem faced by RNNs. LSTMs have feedback connections which make them different from traditional neural nets. This property enables LSTMs to process entire sequences of data without treating each point in the sequence independently, but rather, retaining useful information about previous data in the sequence to help with the processing of new data points. As a result, LSTMs are particularly good at processing sequences of data such as text, speech and general time-series.

Model structure: At a basic level, the output of an LSTM at a particular point in time is dependant on three things:

1) The current long-term memory of the network — known as the cell state.
2) The output at the previous point in time — known as the previous hidden state.
3) The input data at the current time step.

LSTMs use a series of gates which control how the information in a sequence of data comes into, is stored in and leaves the network. There are three gates in a typical LSTM; forget gate, input gate and output gate. These gates can be thought of as filters and are each their own neural network.

Working: The key steps involved in the working of the LSTM gate are described below:

The first step in the process is the forget gate. The previous hidden state and the new input data are fed into a neural network. This network generates a vector where each element is in the interval $[0, 1]$. This network (within the forget gate) is trained so that it outputs a value close to 0 when a component of the input is deemed irrelevant and closer to 1 when relevant. These outputted values are then sent up and point-wise multiplied with the previous cell state. By doing this, the forget gate essentially decides which pieces of the long-term memory should now be forgotten given the previous hidden state and the new data point in the sequence.

The next step involves the new memory network and the input gate. The goal of this step is to determine what new information should be added to the networks long-term memory (cell state), given the previous hidden state and new input data. The new memory network is a tanh activated neural network

which has learned how to combine the previous hidden state and new input data to generate a 'new memory update vector'. This vector essentially contains information from the new input data given the context from the previous hidden state. The input gate is a sigmoid activated network which acts as a filter, identifying which components of the 'new memory vector' are worth retaining. This network will output a vector of values in $[0, 1]$. The outputs of new memory network and input gate are point-wise multiplied. The resulting combined vector is then added to the cell state, resulting in the long-term memory of the network being updated.

The final step involves the output gate for computing the new hidden state. We first apply the tanh function to the current cell state to obtain the squished cell state, which now lies in $[-1, 1]$. The previous hidden state and current input data are passed through the sigmoid activated neural network to obtain the filter vector. Finally, this filter vector is applied to the squished cell state by point-wise multiplication to get the new hidden state.

LSTMs generally give rise to models which are more sophisticated and accurate as compared to ARIMA models. However, one key requirement for LSTM models is that there should be an abundance of training data available. This would mean that such an analysis is possible only for long-term trading and not for short-term trading.

IV. Modelling

In this section, we apply ARIMA and LSTM models to the problem at hand for making forecasts. As discssed in section 2, we will be focusing on the Infosys stock as this was found to have highest returns with minimal risk.

We take a look at the plot for stock price of Infosys again. The plot is shown in Figure 17. On visualizing the stock, it is evident that the prices are not stationary and possess some integrating effect. We examine this through the ACF(Auto correlation) and PACF(Partial Auto correlation) plots.

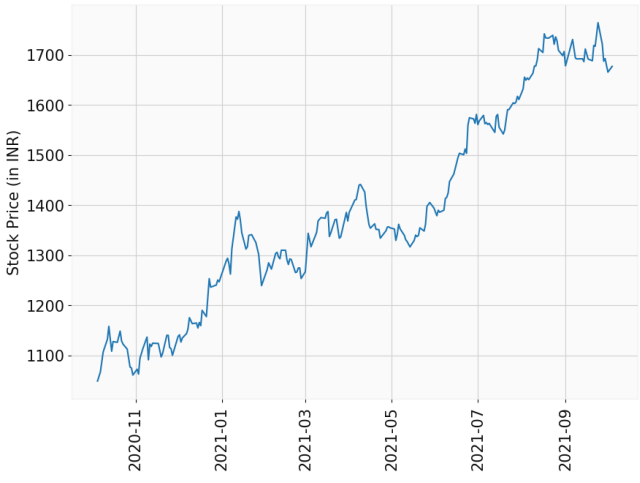

Fig. 17. Stock price chart for Infosys

These plots are shown in Figures 18 and 19. ACF plot shows long memory trend and PACF plot suggests a strong relationship between $v[k]$ and $v[k-1]$. This means that the signal could be integrating of order 1. To confirm this, ADF test was used.

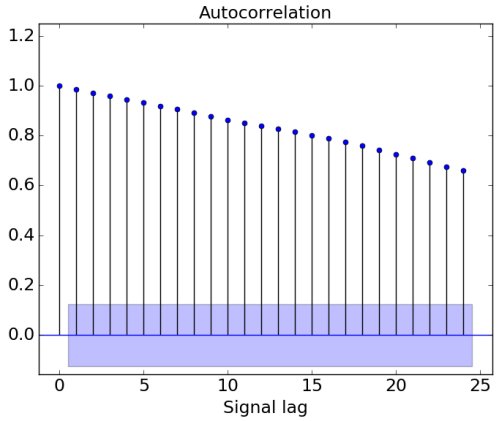

Fig. 18. Autocorrelation for prices

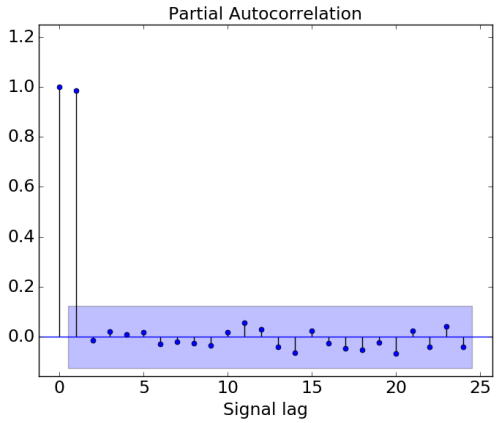

Fig. 19. Partial Autocorrelation for prices

On differencing, we obtain the plots as shown in Figures 20 and 21. The ACF and PACF clearly shows zero MA and AR processes. Hence:

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

is the modelling choice for our Infosys stock. The residuals of the differenced series, plotted in Figure 22, shows white noise characteristics and thus further strengthens our model assumption.

The time series model we derived just now was used for fitting the stock price data till Oct-2020. The prices for the remaining time was forecasted using the fitted model and the prediction was compared with the true stock prices. The plot showing the predictions is given in Figure 23. The grey shaded region around the prediction represents the confidence interval for the predictions, obtained through statistical analysis.

We see that the model predicts the uptrend accurately and that the true values lie within the confidence interval. However, though the model predicts the error accurately, the model

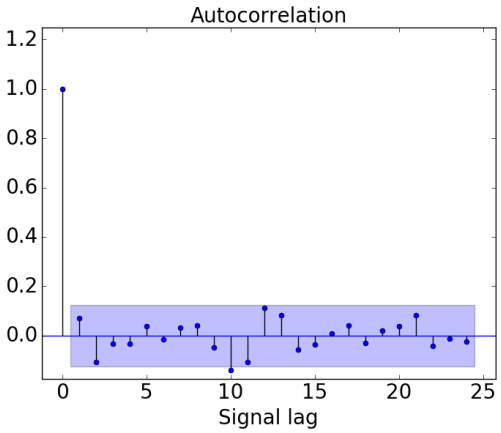

Fig. 20. Autocorrelation for difference of prices

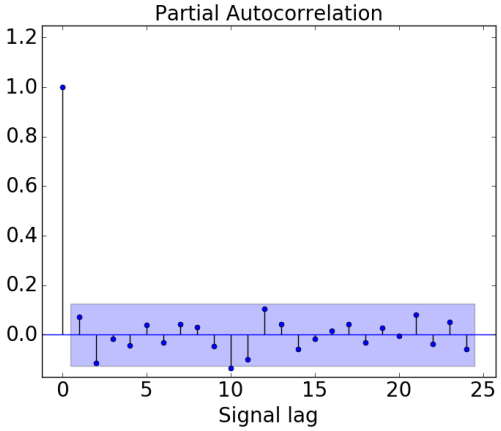

Fig. 21. Partial Autocorrelation for difference of prices

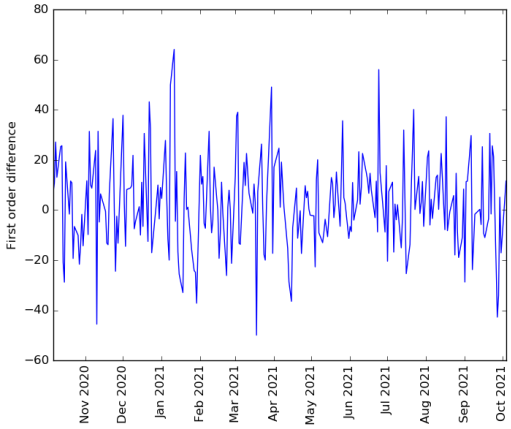

Fig. 22. First order differencing

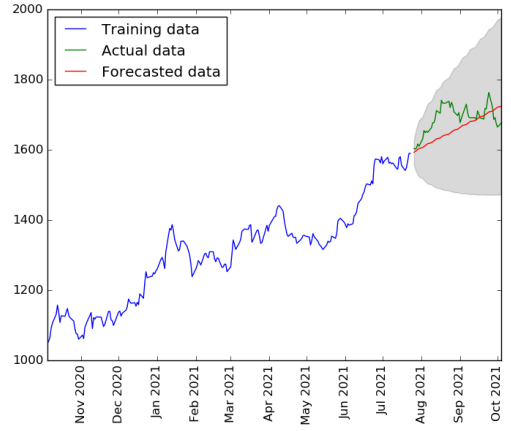

Fig. 23. Forecasting of Infosys prices using ARIMA model

seems to be too simple to accurately model the complexity associated with making the predictions.

In this context, we make use of the more sophisticated LSTM model. This model will yield much better results compared to traditional time series analysis and could capture non-linearity in a better fashion. The output from the model is shown in Figure 24. The neural network clearly outperforms the ARIMA model. However, it must be noted that training the LSTM model generally requires a larger amount of training data. Furthermore, confidence intervals cannot be generated in general for LSTM-based networks.

V. Conclusions

Based on our extensive analysis so far regarding the financial dataset, we arrive at the following set of conclusions:

- A lot of information regarding the dataset can be obtained through simple exploratory analysis. It might be worthwhile to spend time understanding the data rather than blindly applying algorithms on it.
- Exploratory analysis of the given set of stocks revealed that, of the lot, Infosys was the one with rather high gain despite having low risk. Of course, one may choose other stocks depending on their goals and risk-appetite, but from a minimal risk perspective, Infosys is a good choice.
- Real life problems such as stock market analysis could be cleverly formulated as time series analysis problems by making use of traditional time series analysis concepts.
- Classical time series analysis methods like ARIMA as well as more modern approaches like LSTMs could be used for modelling time series data. The theoretical background for these models were examined thoroughly.
- Both the ARIMA model as well as the LSTM model were able to forecast the prices for Infosys stock. ARIMA, despite being a rather simple model, predicted the uptrend accurately whereas LSTM was able to predict the price much more accurately thanks to its sophisticated nature.

VI. Avenues for further research

Only two models. namely ARIMA and LSTM were explored in this analysis. Several other tools exist in literature for the analysis of time series data. Trying out more models might help achieve better results. Furthermore, rather than considering only historical price data, effect of influence of

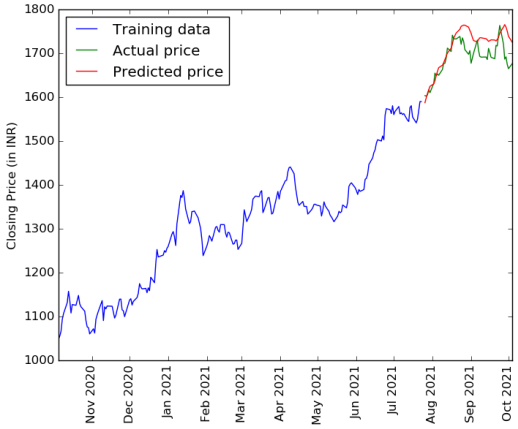

Fig. 24. Forecasting of Infosys prices using LSTM model

other factors in forecasting the prices of stocks could also be studied to give a more comprehensive picture of the scenario.

REFERENCES

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[5] "Time series." https://en.wikipedia.org/wiki/Time_series.

[6] "Long short term memory." https://en.wikipedia.org/wiki/Long_short-term_memory.

A mathematical essay on linear regression

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this assignment is to explore the mathematical formalism behind linear regression and then to use it in a real-life application. In this assignment, as a real-life application, linear regression is used to formally identify the relationship between socioeconomic status and cancer incidence, mortality rates. Linear regression is implemented using Python. The analysis enables us to arrive at the conclusion that the socioeconomic status does indeed have an impact on the cancer incidence. mortality rates. This is a reworked version of the original assignment with improvements to the Modelling section in the form of more comprehensive statistical analysis of the models. All of the plots were also modified to improve clarity.**

***Index Terms*—linear regression, python, visualization**

I. Introduction

Cancer is one of the most pressing problems faced by society today. While there exists methods to prevent as well as cure cancer, question remains as to whether everyone has easy access to it. It is thus necessary to systematically analyse the impact the socioeconomic status of an individual has on their chance of cancer incidence/ mortality. The goal here is therefore to make use of data to see how cancer incidence/mortality rates of a person is influenced by their income/societal status.

With the technological advancements we have made, it is now possible to make use of data analytic tools to derive insights from data. By aggregating, cleaning and analysing large amounts of data, it is possible to draw inferences regarding how some parameters may influence others. Linear Regression is one such model which can be used for building linear relationships between variables. The attempt is to make use of Linear Regression to see how various factors affect cancer incidence/mortality of a population.

In this work, we make use of Python for performing the necessary data manipulations. Various libraries in Python are used for cleaning the data, visualizing the data and analyzing the data. Namely, the libraries pandas, seaborn and scikit-learn are extensively used. Jupyter notebook environment is used to easily integrate concise explanation for the code along with the code itself.

After a careful analysis of the available data, and using linear regression, we arrive at the conclusion that the socioeconomic status of an individual does have a significant impact on his chance of cancer incidence/ mortality. More specifically, we find that poverty rate in general has a positive correlation with cancer incidence/ mortality, meaning as poverty rate increases there is a higher chance for cancer incidence/ mortality in that population. We also find that the median income has a negative correlation with cancer incidence/ mortality. Thus, it can be said that low-income groups are at a greater risk of suffering from cancer and it is imperative to address the issue to ensure well being of all members of the society.

Section II gives an overview of the various techniques used for data cleaning and an initial exploratory analysis. A lot of insights can be gained just by making qualitative observations from the given data. Section III gives a short description of the mathematical formalism behind linear regression. section IV sates the various results that are obtained by applying linear regression in this particular case. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we describe the process of data cleaning and visualization.

The given dataset consists of 3134 rows and 25 columns. A brief overview of the dataset is presented in Figure 1.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3134 entries, 0 to 3133
Data columns (total 25 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   State              3134 non-null   object
 1   AreaName           3134 non-null   object
 2   All_Poverty        3134 non-null   int64
 3   M_Poverty          3134 non-null   int64
 4   F_Poverty          3134 non-null   int64
 5   FIPS               3134 non-null   int64
 6   Med_Income         3133 non-null   float64
 7   Med_Income_White   3132 non-null   float64
 8   Med_Income_Black   1924 non-null   float64
 9   Med_Income_Nat_Am  1474 non-null   float64
 10  Med_Income_Asian   1377 non-null   float64
 11  Hispanic           2453 non-null   float64
 12  M_With             3134 non-null   int64
 13  M_Without          3134 non-null   int64
 14  F_With             3134 non-null   int64
 15  F_Without          3134 non-null   int64
 16  All_With           3134 non-null   int64
 17  All_Without        3134 non-null   int64
 18  fips_x             3134 non-null   int64
 19  Incidence_Rate     3134 non-null   object
 20  Avg_Ann_Incidence  3134 non-null   object
 21  recent_trend       3134 non-null   object
 22  fips_y             3134 non-null   int64
 23  Mortality_Rate     3134 non-null   object
 24  Avg_Ann_Deaths     3134 non-null   object
dtypes: float64(6), int64(12), object(7)
memory usage: 636.6+ KB
```

Fig. 1. Summary of the raw dataset

Our first task is to weed out columns which are not of relevance. Since our objective is to make inferences for the whole population and not to make inferences for various states/areas

separately, discerning between states/areas is unnecessary. This renders columns State, AreaName, FIPS, fips_x and fips_y. Hence, we drop these columns.

Next, we observe that the columns Incidence_Rate, Avg_Ann_Incidence, Mortality_Rate and Avg_Ann_Deaths are all marked as having "object" datatype. This is an indication that not all values in these columns are numerical. This will present a problem during mathematical analyses. Hence, it is essential to examine these columns and try to make all values numerical or categorical.

On examination, we find the that there are some non-numerical entries in each of these columns. The type of entry and method adopted to convert it to a numerical value are discussed below:

- * - This indicates that the actual value is very low. It seems reasonable to replace these entries using 0. However, this results in the concentration of a lot of values at 0 and severely impedes the performance of regression models. Hence, these rows were removed.
- entries ending with # - This is just an error in formatting. The problem can be resolved by simply removing the # symbol.
- _ and __ - These indicate lack of information. Although there are imputation strategies available to deal with such data, we do not apply them here. This is because of two reasons:
  - Missing data is present in variables like Incidence_Rate which is what we would like to predict from other parameters. In this sense, these are like target variables and hence imputation seems non-ideal.
  - The rows with missing data for these variables accounts for only about 6% of the total number of rows. Since this is a small fraction, there isn't really a lot of loss of data.

After performing all these operations, we are left with a cleaned dataset, whose summary is presented in Figure 2.

At this point, we also make an important observation regarding the variables Incidence_Rate, Avg_Ann_Incidence, Mortality_Rate and Avg_Ann_Deaths. Avg_Ann quantities represent the **numbers for the entire population** , whereas Incidence_Rate and Mortality_Rate represent the **numbers normalized using the total population.** Since what is of relevance to us is the normalized rate, we shall be **considering only Incidence_Rate and Mortality_Rate in all further analysis.**

Our objective is to see the impact of the following two factors on cancer incidence and mortality:

1) Economic status
2) Social status

A. *Economic status*

From the available dataset, we make the following observations:

- The variables we have with respect to the economic status are the following:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2640 entries, 0 to 2639
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   All_Poverty        2640 non-null   int64
 1   M_Poverty          2640 non-null   int64
 2   F_Poverty          2640 non-null   int64
 3   Med_Income         2640 non-null   float64
 4   Med_Income_White   2640 non-null   float64
 5   Med_Income_Black   1818 non-null   float64
 6   Med_Income_Nat_Am  1295 non-null   float64
 7   Med_Income_Asian   1278 non-null   float64
 8   Hispanic           2127 non-null   float64
 9   M_With             2640 non-null   int64
 10  M_Without          2640 non-null   int64
 11  F_With             2640 non-null   int64
 12  F_Without          2640 non-null   int64
 13  All_With           2640 non-null   int64
 14  All_Without        2640 non-null   int64
 15  Incidence_Rate     2640 non-null   float64
 16  Avg_Ann_Incidence  2640 non-null   float64
 17  recent_trend       2640 non-null   category
 18  Mortality_Rate     2640 non-null   float64
 19  Avg_Ann_Deaths     2640 non-null   float64
dtypes: category(1), float64(10), int64(9)
memory usage: 394.8 KB
```

Fig. 2. Summary of the processed dataset

  - Number of individuals below poverty line
  - Median income of individuals
  - Number of individuals who have health insurance
- We expect a **positive correlation between incidence/mortality rate and poverty rate, whereas a negative correlation between median income and incidence/mortality rate.**
- In the case of **health insurance**, we **do not expect it to have much impact on the incidence rate** itself, whereas we expect the number of individuals who are insured (normalized with population) to have some **negative correlation with mortality rate.**

On calculating, we observe that **correlation between All_Poverty, which is the number of individuals below poverty line, and Incidence_Rate is -0.027 which is almost 0!** Does this mean there is no correlation between the two? The important fact to note here is that All_Poverty is the total number of individuals below the poverty line, i.e. it has not been normalized with total population, whereas Incidence_Rate is already normalized. Hence, it is **necessary to normalize this quantity as well with respect to total population.**

We can approximate the total population as the sum of all people with health insurance and all people without health insurance. We also calculate male and female population separately. We add these quantities as separate columns and then compute the **Poverty rate**, which is the number of individuals below poverty lane normalized using population, separately for males and females as well as for the entire population.

After calculating the poverty rates, we calculate the correlation between poverty rate (across all individuals) and incidence rate, mortality rate. We observe correlation values of **0.259** and **0.285**, which indicate **significant positive correlation.** To visualize these observations, we make scatter plots for

incidence rate and mortality rate against poverty rate. These plots are presented in Figure 3 and Figure 4.

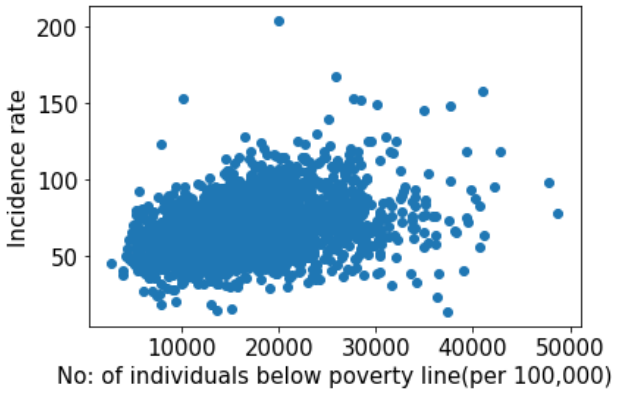

Fig. 3. Cancer incidence v/s Poverty

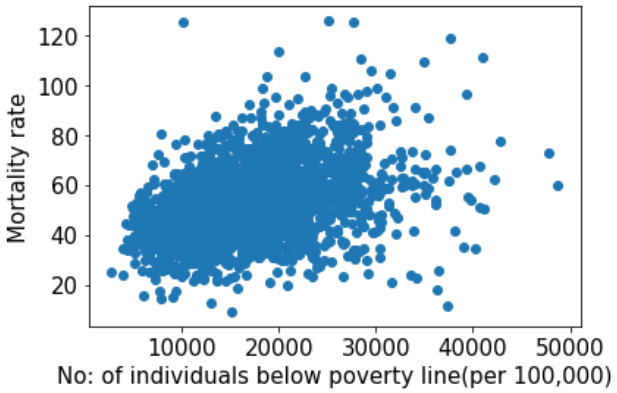

Fig. 4. Mortality v/s Poverty

From the plots, it is clear that cancer incidence/mortality increases, in general, with increase in poverty.

Next, we observe the relationship between median income and cancer incidence/mortality rate. On calculating, we find that the correlation values are **-0.256** and **-0.276** which indicate **significant negative correlation.** To visualize these observations, we make scatter plots for incidence rate and mortality rate against median income. These plots are presented in Figure 5 and Figure 6.

From the plots, it is clear that cancer incidence/mortality decreases, in general, with increase in income.

From the correlation values and plots, it can be qualitatively concluded that as poverty increases/ income decreases, there is a higher chance for cancer incidence/mortality in general.

We will now consider the impact of having a health insurance on cancer incidence/ mortality. On calculating, we observe correlation values of **-0.038** and **-0.124** for incidence and mortality respectively. It seems like there **isn't any correlation at all between cancer incidence and having a health insurance.** This is to be expected because a health insurance is of use only in the treatment of a disease; not so much for its prevention. Unlike the case of cancer incidence, there seems to be a **significant negative correlation between having a health insurance and mortality rate.** This is illustrated in Figure 7. Again, this is to be expected since having a health insurance provides an incentive to individuals to seek appropriate treatment without considering financial limitations.

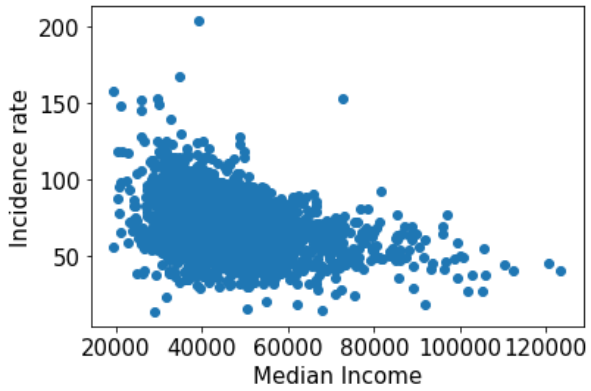

Fig. 5. Cancer incidence v/s Income

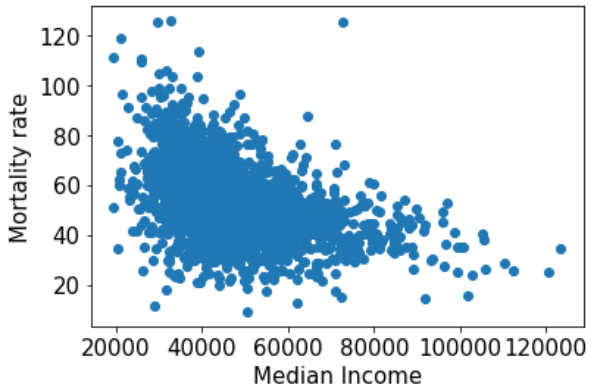

Fig. 6. Mortality v/s Income

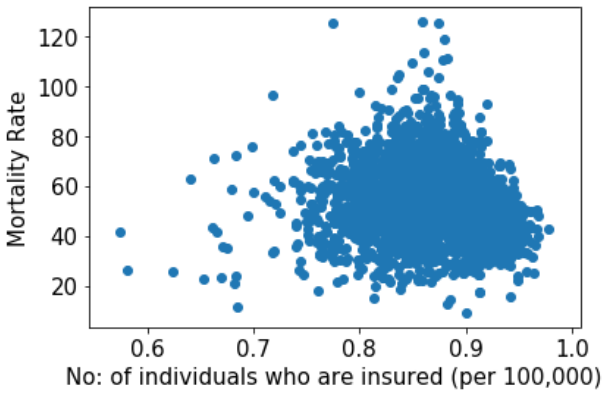

Fig. 7. Mortality v/s Insurance

B. Social status

Our target is to associate social status with cancer incidence/ mortality. Towards this, we shall consider the following two broad social classification criterion:

- Gender
- Ethnicity

The major challenge we face in this case is that we **do not have direct access to incidence/mortality rate for population subsections separately.** For instance, we do not know the separate count of cancer incidence of male individuals and cancer incidence of female individuals. This prohibits any direct comparison. As a result, it is necessary to make an **indirect comparison** based on the available parameters.

We first consider gender. For gender, we consider the poverty rate for males and females separately. Since we have already concluded from earlier analysis that poverty rate has a positive correlation with both incidence rate and mortality rate, **if we can conclude that one section has a higher poverty rate in general, we can conclude that this section suffers from higher incidence/mortality as well.**

To see how the poverty rates of the two sections compare, we make use of a box plot. The box plot is shown in Figure 8. From the plot, we can **qualitatively see that females have higher poverty rate, in general.** To make this more quantitative, consider Figure 9. **Clearly, mean and median of poverty rate of females are higher than males.** Hence, from this, it can be estimated that female population in general has higher poverty rate and **consequently higher values for cancer incidence and mortality rate.**

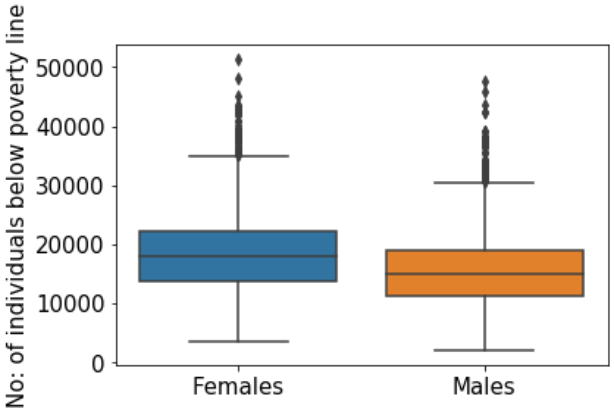

Fig. 8. Gender v/s Poverty

Next, we explore the relationship between ethnicity and cancer incidence/mortality. In this case, the parameter available to us in median income. As we have seen before, a lower median income would correspond to higher cancer incidence/mortality. To see how the median income varies among different ethnic groups, we again make use of a box plot. The plot is given in Figure 11. It can be seen that the median income of different sections are different, with some sections having higher values and others having lower. A quantitative description is given in

	F_Poverty_Rate	M_Poverty_Rate
count	2640.000000	2640.000000
mean	18438.267600	15449.460381
std	6671.494646	6015.827048
min	3422.382671	1972.637607
25%	13757.330077	11153.223760
50%	17891.631556	14869.535936
75%	22205.200648	18859.267008
max	51264.842540	47576.177285

Fig. 9. Statistics for poverty distribution

Figure 12. It can be seen that the mean of the median income follow the following order for ethnic groups: Asians $>$ Whites $>$ Native Americans $>$ Hispanics $>$ Blacks. Hence, from this, it can be estimated that cancer incidence/mortality rate also follows the same order for ethnic groups.

It must be noted that in all the analysis we have done in this subsection, we have attempted to correlate social status with economic indicators like poverty and income. This approach, though insightful, cannot guarantee a completely accurate analysis, since for this we will need information regarding incidence/mortality rate of each section of the population separately.

C. Incidence v/s Mortality

Finally, for the sake of completion, we explore the relationship between cancer incidence and mortality. This is fairly obvious, as we expect a high positive correlation between the two. Visual evidence is presented in Figure 10. We observe a correlation of **0.867** It is also worthwhile to note here that in general as one increases the other also increases with the only exception being the case of insurance rate, which has already been discussed.

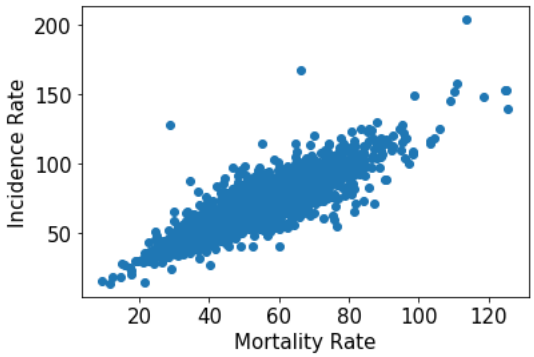

Fig. 10. Incidence v/s Mortality

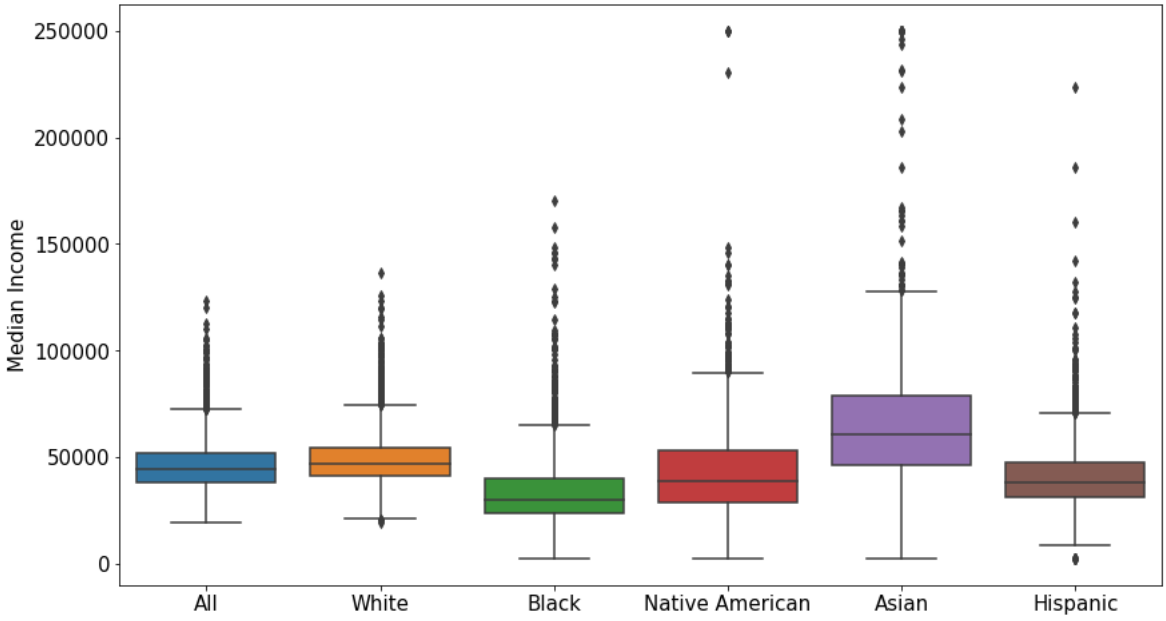

Fig. 11. Ethnicity v/s Median income

	Med_Income	Med_Income_White	Med_Income_Black	Med_Income_Nat_Am	Med_Income_Asian	Hispanic
count	2640.000000	2640.000000	1818.000000	1295.000000	1278.000000	2127.000000
mean	46542.398485	49386.191667	34708.625413	43468.138224	65998.231612	41039.206394
std	12529.141437	12713.167597	18087.803105	24197.024884	34411.305663	16272.038982
min	19328.000000	19340.000000	2499.000000	2499.000000	2499.000000	2499.000000
25%	38226.250000	41175.250000	23835.000000	28969.000000	46340.500000	31591.500000
50%	44497.000000	47043.500000	29983.500000	39028.000000	60516.000000	38198.000000
75%	51993.250000	54484.750000	40337.250000	53190.500000	79002.000000	47282.500000
max	123453.000000	136311.000000	170195.000000	250001.000000	250001.000000	223750.000000

Fig. 12. Statistics for income distribution

III. Model: Linear Regression

In this section, we will give a brief overview of the mathematical formalism behind the linear regression model.

Regression is a statistical technique where both the dependent and independent variable takes continuous values and a model is fitted on the explanatory variables. Linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables. In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data.

It must be noted that in general, linear regression is used for creating a linear model of existing data and then using this to predict values for scenarios in which the target variables is unknown. However, in our particular use case, rather than attempting to do this, we will make the model and then use it on the same data; the idea is to see how well the model models the proposed relationship between the variables. A general mathematical description of linear regression is presented below.

Given a data set $\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear. This relationship is modeled through a disturbance term or error variable ϵ — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta} + \varepsilon_i,$$

for $i = 1, \ldots, n$ and where T denotes the transpose, so that $\mathbf{x_i^T}\boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x_i}$ and $\boldsymbol{\beta}$.

Often these n equations are stacked together and written in matrix notation as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} \mathbf{x}_1^\mathsf{T} \\ \mathbf{x}_2^\mathsf{T} \\ \vdots \\ \mathbf{x}_n^\mathsf{T} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

The goal, in general ,is to compute the vector $\boldsymbol{\beta}$ when given $\mathbf{y}$ and $\mathbf{X}$. For the case of simple linear regression, also known as Ordinary Least Squares (OLS), there exists a closed form solution for the following optimization problem, in which we find $\hat{\boldsymbol{\beta}}$ such that:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{X}\boldsymbol{\beta} - \mathbf{y}||^2$$

The closed form solution is given by:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X^T X})^{-1}\mathbf{X^T y}$$

After computing a solution, it is necessary to have some sort of benchmark to measure the quality of the solution. In our use case, we would like to know how well the chosen parameter represents the target variable. Towards this, we make use of the following two metrics:

1) **Mean Squared Error (MSE)**: MSE is the mean of squares of the error terms, i.e., MSE is given by:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where y_i is the actual value and $\hat{y}_i$ is the predicted value. Note that this is an absolute metric, meaning the value of MSE cannot be compared across different datasets but can be used for comparison among different models of the same problem.

2) **Coefficient of determination (R^2)**: R^2 is the proportion of the variation in the dependent variable that is predictable from the independent variable(s). It is computed as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value and $\bar{y}_i$ is the sample mean given by:

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$

TABLE I
BENCHMARKS FOR MODEL 1 (ONLY POVERTY RATE USED)

Type	Incidence Rate	Mortality Rate
MSE	279.397	169.968
R^2	0.107	0.147

TABLE II
BENCHMARKS FOR MODEL 2 (ONLY MEDIAN INCOME USED)

Type	Incidence Rate	Mortality Rate
MSE	267.633	159.726
R^2	0.144	0.198

Unlike MSE, R^2 is a relative measure and hence is somewhat data agnostic. Typically, R^2 values lies in between 0 and 1, with higher values signifying a better fit.

IV. MODELLING

In this section, we discuss the application the linear regression model to our problem.

Our primary goal is to identify how cancer incidence and mortality rates are affected by income. More specifically, we would like to see if low-income groups are more prone to cancer and subsequent fatality. The two main parameters we can consider for analysing the income of a particular group are poverty rate and median income.

We employ linear regression keeping cancer incidence rate and mortality as the target variables. We consider the following four models, each taking different input variables:

1) Model 1: Poverty rate as the only input variable
2) Model 2: Median income as the only input variable
3) Model 3: Both poverty rate and median income as input variables
4) Model 4: Polynomial features generated from poverty rate and median income as input variables

A. Model 1

We obtain the plot shown in Figure 13 for incidence rate and plot shown in Figure 14 for mortality rate. The values of bench-marking parameters are summarised in Table I.

B. Model 2

We obtain the plot shown in Figure 15 for incidence rate and plot shown in Figure 16 for mortality rate. The values of bench-marking parameters are summarised in Table II.

It can be seen that we obtain **better fit, in general by using median income instead of poverty rate.** This can be seen in terms of lower MSE values and higher R^2 scores.

C. Model 3

In this case, we give both poverty rate and median income as input paramters. Since, we have two input parameters, it is not possible to visualize the plot in 2D. We can still however obtain the bench-marking parameters. The values of bench-marking parameters are summarised in Table III.

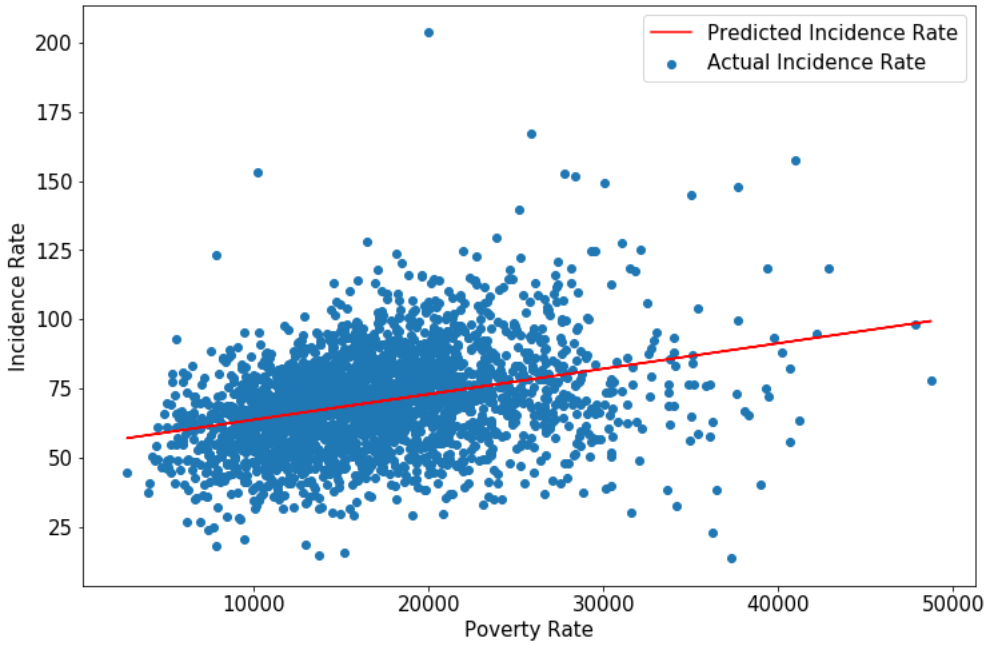

Fig. 13. Incidence prediction from poverty rate

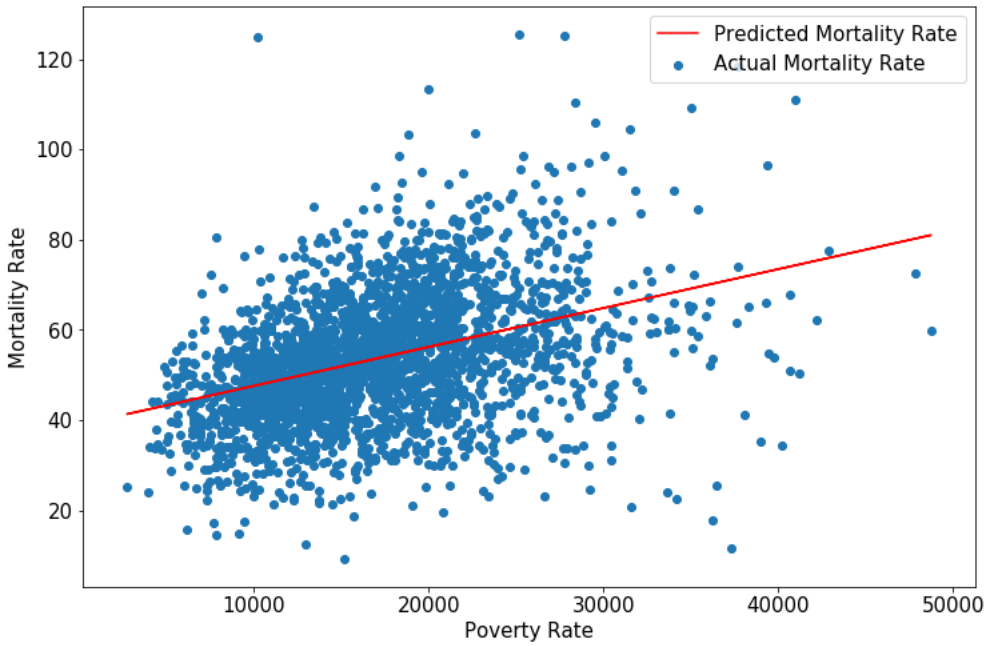

Fig. 14. Mortality prediction from poverty rate

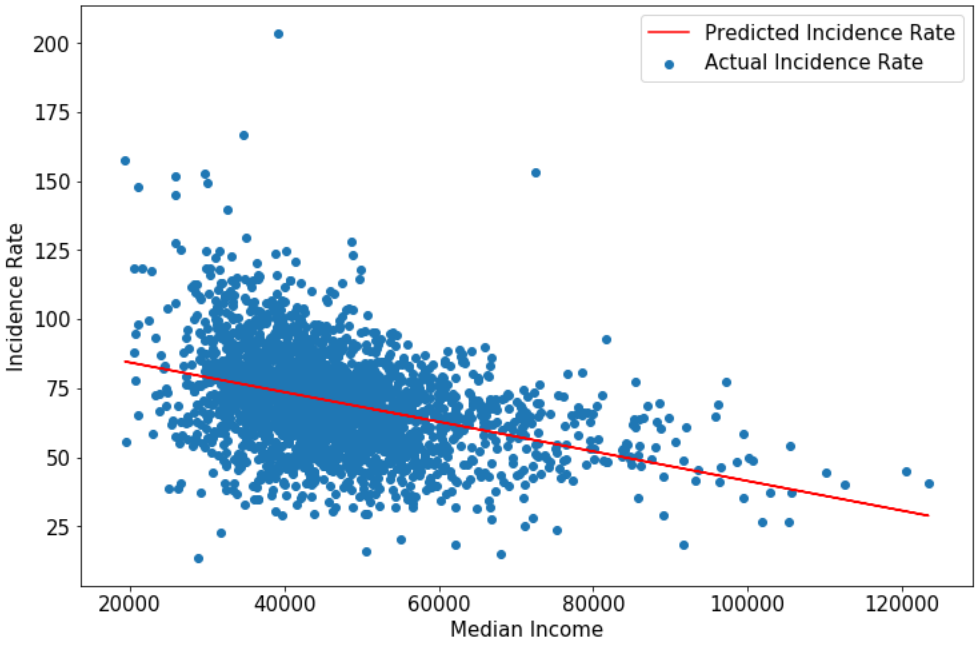

Fig. 15. Incidence prediction from median income

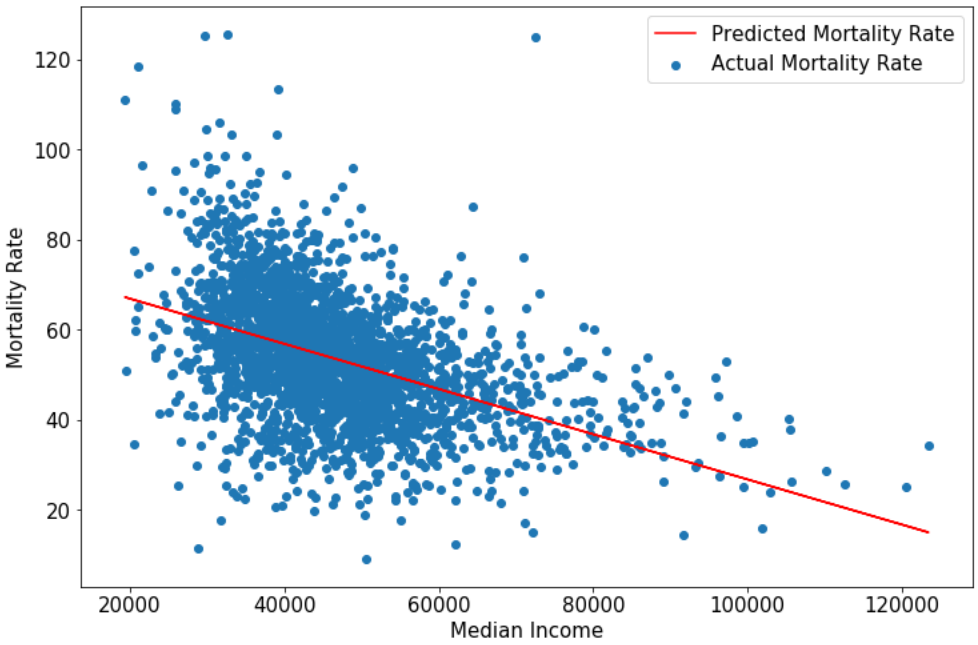

Fig. 16. Mortality prediction from median income

TABLE III
BENCHMARKS FOR MODEL 3 (INCOME + POVERTY)

Type	Incidence Rate	Mortality Rate
MSE	267.032	159.186
R^2	0.146	0.201

TABLE IV
BENCHMARKS FOR MODEL 4 (POLYNOMIAL FEATURES)

Type	Incidence Rate	Mortality Rate
MSE	249.674	146.292
R^2	0.202	0.266

It can be seen that we get a better fit than model 2, though only by a marginal amount, again **signifying that majority of the contribution is from the median income parameter.**

D. Model 4

In this case, we generate a polynomial of degree 6 from the combination of parameters poverty rate and median income. Again, since we have multiple input parameters, it is not possible to visualize the plot. We can still however obtain the bench-marking parameters. The values of bench-marking parameters are summarised in Table IV.

It can be seen that we get a **better fit than model 3, by a significant amount.** This makes sense intuitively because it is not necessary that cancer incidence/mortality is a linear function of poverty rate and median income, but rather can depend on higher powers of these parameters.

Statistical evidence

While analyzing results from a linear regression model, it is often a good idea to analyze the statistical properties of the model. A thorough statistical analysis can provide us more insights regarding whether the results from the model are statistically significant or not. The analysis of the statistical significance can be carried out using two metrics: P-value and F-statistic. These metrics are discussed below:

1) **P-value:** The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value (< 0.05) indicates that you can reject the null hypothesis. In other words, a predictor that has a low p-value is likely to be a meaningful addition to the model because changes in the predictor's value are related to changes in the response variable. Conversely, a larger (insignificant) p-value suggests that changes in the predictor are not associated with changes in the response.
2) **F-statistic:** F-statistic is an indicator of a relationship between dependent variable and the response variables. Because individual t-tests assume that each variable comes from an independent distribution, it might lead to errors as this leads to multi-collinearity issues and also accumulation of Type-1 error. Hence, instead we use by the F-statistic to conclude if there is a significant relationship between dependent variables and the response variable. The larger the F statistic, the better it is.

TABLE V
STATISTICAL EVIDENCE FOR THE MODELS

Metric	Incidence Rate	Mortality Rate
Model 1		
F-statistic	314.9	454.1
P-value	0.000	0.000
Model 2		
F-statistic	444.7	652.4
P-value	0.000	0.000
Model 3		
F-statistic	225.7	331.7
P-value (Poverty)	0.015	0.003
P-value (Income)	0.000	0.000

The metrics for the models are presented in Table V. We do not consider model 4 separately, since the features are essentially same as that of model 3. Clearly, the **F-statistic is very high for all the models**, indicating that the models are all valid. The p-value is also smaller than 0.05 for all the models. However, it can be seen that the **p-value is relatively high for the poverty metric in model 3.** This is something we expect intuitively, as we have already seen that the majority of the contribution in model 3 comes from the median income parameter and that the contribution from the poverty parameter is small. This has now been formally established through p-values.

V. CONCLUSIONS

From the four models we have built, it is **unambiguously clear that cancer incidence and mortality are very much impacted by the socioeconomic status.** By using a mathematical model, it has been formally proved that there exists a **positive correlation** between poverty rate and cancer incidence/mortality whereas there exists a negative correlation between median income and cancer incidence/mortality. The strength of the correlation has also been **quantified through bench-marking parameters, namely MSE and $\mathbf{R^2}$.**

It has also been inferred that **gender and ethnicity play a role in determining the economic status** of an individual as evidenced by the poverty rate and median income distributions. It has also been observed that individuals **not having a health insurance are at a greater risk of suffering death from cancer as compared to individuals who are insured.**

From the above listed conclusions, it is abundantly clear that **making health-related policies specifically directed at the low-income sections of the population** is the need of the hour. Focus should be on providing accessible and affordable health insurance as well as treatment to all sections of the population, irrespective of ethnicity and gender.

VI. AVENUES FOR FURTHER RESEARCH

Although inferences were made regarding how social status affects cancer incidence/mortality, since separate values for

cancer incidence/mortality rates for different ethnic/gender sections were not available, direct predictions were not made. Rather, predictions were made indirectly using parameters like median income and poverty. If data can be collected separately for different ethnic/gender groups, a lot more insights can be drawn. Also, there are probably several other parameters which influence cancer incidence/ mortality. A detailed study of these parameters may also be a worthile avenue for further exploration.

REFERENCES

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[5] "Linear regression." https://en.wikipedia.org/wiki/Linear_regression.

A mathematical essay on logistic regression

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this assignment is to explore the mathematical formalism behind logistic regression and then to use it in a real-life application. In this assignment, as a real-life application, logistic regression is used to formally identify the factors which could have been used to predict the chance of survival of an individual in the historical sinking of The Titanic. Data visualization, cleaning and modelling is done using Python. The analysis enables us to arrive at the conclusion that although luck played a major role in predicting survival, other factors like socioeconomic status and gender does indeed have an impact on the chance of survival. This is a reworked version of the original assignment with improvements to the Modelling section in the form of enhancements to the vanilla logistic regression model.**

***Index Terms*—logistic regression, python, visualization, predictive modelling**

I. Introduction

The sinking of Titanic is one of the most tragic incidents in the history of humanity. On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg, resulting in the death of 1502 out of the total 2224 individuals (including passengers and the crew). Although there definitely was some element of luck involved in determining who survived and who did not, it is necessary to examine if any other factors(such as socioeconomic status) played a role in determining the survival of an individual. This can help in bringing out involuntary biases which may play a role in imposing disadvantages on certain sections of the society.

In this assignment, the goal is to make use of logistic regression to tackle the above mentioned problem. Logistic regression is a popular mathematical model for estimating how a binary variable is influenced by a variety of known factors. By performing logistic regression analysis, we will be able to get an idea of how each feature influences the target variable. This model can then be used for predicting the target variable from the known factors for cases in which the actual value of the target variable is unknown.

In this particular problem setting, the goal is to make use of a logistic regression model for the particular case of predicting whether an individual will survive the sinking of Titanic, provided we know certain characteristics of the individual, including but not limited to socioeconomic status and gender. By making use of data for which we know if an individual survived or not, we then build a model for the data for which we do not know this. The accuracy of the model can then be used to assess how good the identified relationships are.

Section II gives an overview of the various techniques used for data cleaning, feature engineering and an initial exploratory analysis. A lot of insights can be gained just by making qualitative observations from the given data. Section III gives a short description of the mathematical formalism behind logistic regression. Section IV sates the various results that are obtained by applying logistic regression in this particular case. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we describe the process of data cleaning, feature engineering and data visualization.

A. Data Cleaning

We are given two datasets: a training dataset and a test dataset. The training dataset consists of 891 rows and 11 columns, whereas the test dataset consists of 418 rows and 10 columns. A brief overview of the training dataset is presented in Figure 1 and of the test dataset is presented in Figure 2. We see that we know the ground truth values for whether the individual survived or not in the training dataset whereas we do not know this for the test dataset. We will keep the test dataset aside and use it only after we have completed building the model.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

Fig. 1. Summary of the training dataset

Our next task is to account for the missing values in the dataset. We see that there are missing values in three columns: ”Age”, ”Cabin” and ”Embarked”. Each column is dealt with separately and the procedure is described below.

The values in the column ”Age” are plotted in a boxplot in Figure 3. From the plot, it can be seen that the distribution **is**

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 418 entries, 892 to 1309
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Pclass    418 non-null    int64
 1   Name      418 non-null    object
 2   Sex       418 non-null    object
 3   Age       332 non-null    float64
 4   SibSp     418 non-null    int64
 5   Parch     418 non-null    int64
 6   Ticket    418 non-null    object
 7   Fare      417 non-null    float64
 8   Cabin     91 non-null     object
 9   Embarked  418 non-null    object
dtypes: float64(2), int64(3), object(5)
memory usage: 35.9+ KB
```

Fig. 2. Summary of the test dataset

skewed to the right. In such scenarios, it is better to impute the missing values **using median as compared to mean.** Hence the missing values are replaced with the computed median, which happens to be 28.

We see that we only have **202 non-null values for column "Cabin" out of the total 891, which is merely 22.67%.** Since this is so low, we drop this column for predictive modelling. This is further supported by the fact that most of the datapoints in the test set also do not have cabin values. However, we will use the available values to see if there is any correlation between cabin number and chance of survival.

Finally, for the column "Embarked" we have only 2 missing values. Since this is just 2 datapoints, we drop these rows. After handling the missing values, we end up with a dataset which is summarized by Figure 4.

We also note that the feature "Fare" has a missing value in the test set, but not in the training set. We perform mean imputing for this feature in the test set.

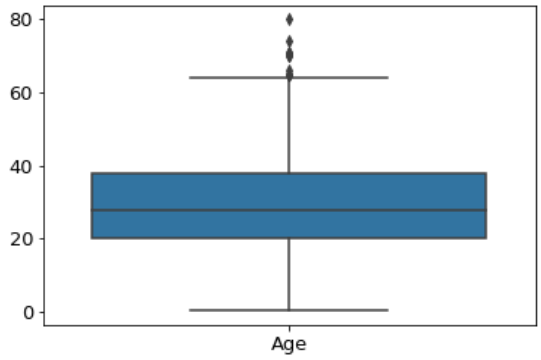

Fig. 3. Distribution of age

B. Data Visualization/ Qualitative Analysis

To get a qualitative picture, we make plots of relevant features against the survival rate, which is the number of individuals who survived divided by the total number of individuals.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  889 non-null    int64
 1   Pclass    889 non-null    int64
 2   Name      889 non-null    object
 3   Sex       889 non-null    object
 4   Age       889 non-null    float64
 5   SibSp     889 non-null    int64
 6   Parch     889 non-null    int64
 7   Ticket    889 non-null    object
 8   Fare      889 non-null    float64
 9   Cabin     202 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.3+ KB
```

Fig. 4. Summary of the processed training dataset

First, we make a barplot of passenger class, given by the column "Pclass". We obtain the plot given in Figure 5. From the plot, it is evident that a higher fraction of passengers belonging to a higher passenger class survived as compared to passengers belonging to a lower passenger class. This could be **due to the fact that higher class tickets were often taken by people who were economically and socially privileged thereby giving them an advantage over the lesser privileged sections in times of such crises.** This also reflects an inherent bias present in the society.

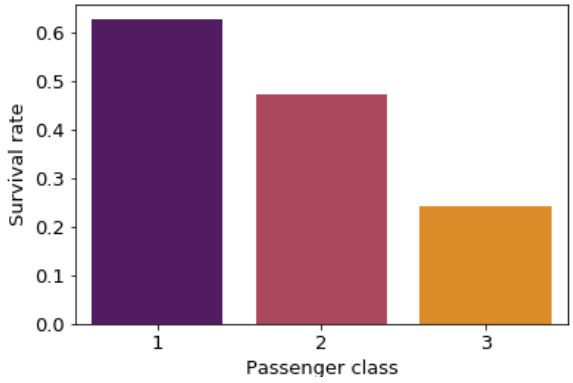

Fig. 5. Survival rate v/s Passenger class

Next, we make a barplot of gender, given by the column "Sex". We obtain the plot given in Figure 6. From the plot, it is evident that the **survival rate is higher for females as compared to males.** This could be because of the social norm of the time, which stipulated giving preference to women and children during times of disasters.

Next, we make a kdeplot (kernel density estimate plot) of age, given by the column "Age". We obtain the plot given in Figure 7. From the plot, we make two observations:

- There was a **greater chance for children to survive.** Again, this could be because priority was given to women

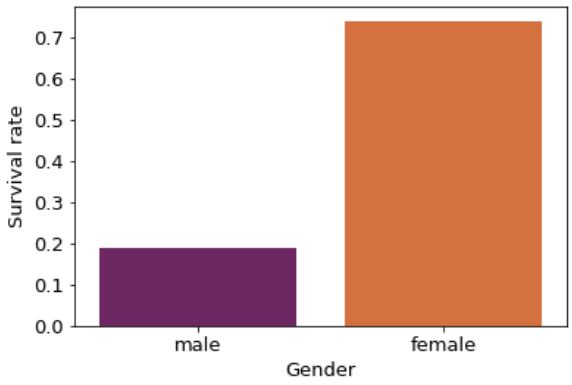

Fig. 6. Survival rate v/s Gender

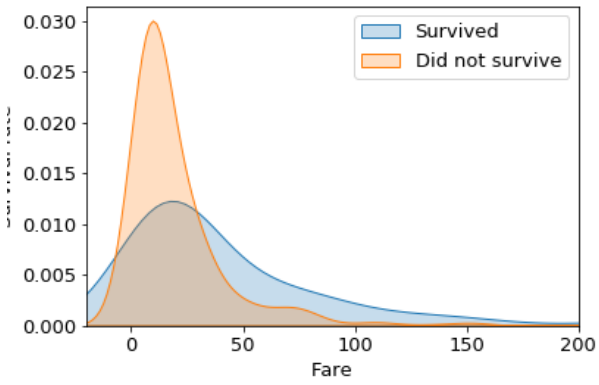

Fig. 8. Survival rate v/s Fare

and children during evacuation.

- **A greater fraction of people in the age group 20-40 did not survive** compared to other age groups; this could be because they stayed back to aid younger and older individuals during evacuation.

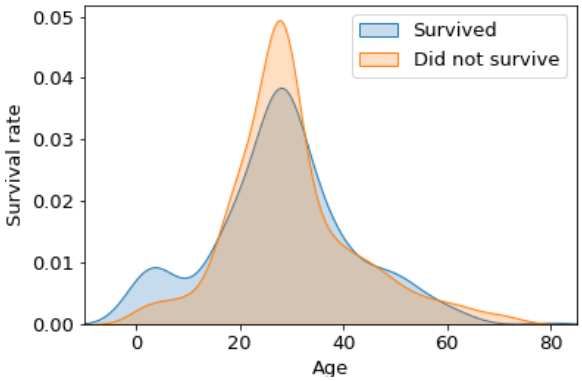

Fig. 7. Survival rate v/s Age

Next, we make a kdeplot (kernel density estimate plot) of fare, given by the column ”Fare”. We obtain the plot given in Figure 8. It can be seen that, in general, **people who paid a higher fare had a higher chance for survival.** This could again be associated with higher social and economic status and the associated privilege, as generally only the richer sections can afford to pay a higher fare.

Next, we explore if there is any relationship between cabin and survival rate. Since considering each cabin separately is impractical, we consider only the cabin group, which is given by the alphabet preceding the number. Note that we do this only for the datapoints for which the cabin information is available. Rest of the datapoints are assigned the cabin group ”U”. We obtain the plot given in Figure 9. Due to the limited data available, it is not possible to make conclusive statements regarding the impact of cabin group on survival rate. However, if more data is available, it might be possible to correlate the cabin group with survival rate as we intuitively expect certain cabin groups to have more access to lifeboats and other such similar factors.

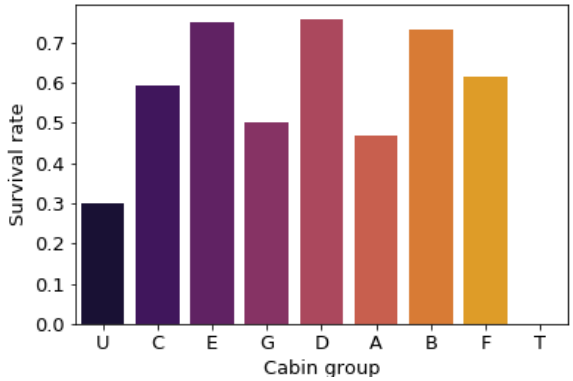

Fig. 9. Survival rate v/s Cabin group

C. Feature Extraction

In this section, we look at manipulating some of the existing features so as to make them more useful.

We do not expect the name of an individual to directly have any influence on his/her survival chance. However, **the honorifics associated with the name might have an impact on survival chance.** Hence, we extract the honorifics and store them in a separate column titled ”title”. We expect this feature to capture the socioeconomic status of an individual.

The categorical features we have now are ”Pclass”, ”Sex”, ”Embarked” and ”title”. Of these, ”Pclass” which denotes the passenger class has a numbering by default; as we have seen before, higher the class of the passenger, higher is the chance of survival. Hence, we maintain the existing numerical ordering for this feature.

For the other features, we implement one-hot encoding. We cannot employ ordinal ordering as this would enforce an ordering of classes which does not naturally exist. Although

in some applications doing one-hot encoding would blow up the number of features, in this case, since we only have a limited number of features, one-hot encoding is feasible. After performing one-hot encoding, we end up with the dataset summarized in Figure 10.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 1 to 891
Data columns (total 25 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  889 non-null    int64
 1   Pclass    889 non-null    int64
 2   Age       889 non-null    float64
 3   SibSp     889 non-null    int64
 4   Parch     889 non-null    int64
 5   Fare      889 non-null    float64
 6   female    889 non-null    int64
 7   C         889 non-null    int64
 8   Q         889 non-null    int64
 9   Capt      889 non-null    int64
 10  Col       889 non-null    int64
 11  Don       889 non-null    int64
 12  Dr        889 non-null    int64
 13  Jonkheer  889 non-null    int64
 14  Lady      889 non-null    int64
 15  Major     889 non-null    int64
 16  Master    889 non-null    int64
 17  Miss      889 non-null    int64
 18  Mlle      889 non-null    int64
 19  Mme       889 non-null    int64
 20  Mr        889 non-null    int64
 21  Mrs       889 non-null    int64
 22  Ms        889 non-null    int64
 23  Rev       889 non-null    int64
 24  Sir       889 non-null    int64
dtypes: float64(2), int64(23)
memory usage: 212.9 KB
```

Fig. 10. Summary of dataset after feature extraction

III. Model: Logistic Regression

In this section, we will give a brief overview of the mathematical formalism behind the logistic regression model.

Logistic Regression (also called Logit Regression) is a model commonly used to estimate the probability that an instance belongs to a particular class out of K given classes. The most common version is the one with K = 2, in which case the classification is binary. If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled "1"), and otherwise it predicts that it does not (i.e., it belongs to the negative class, labeled "0").

Logistic Regression is a linear model for classification, which means the decision surfaces are linear functions of the input vector x. The simplest case is to model $y(\mathbf{x}) = \mathbf{w^T x} + w_0$ 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

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

returns the posterior probabilities. Decision surfaces correspond to $y(x) =$ constant , which implies $\mathbf{w^T x} + w_0 =$ constant for one to one f, which makes the decision boundary linear. An example of such a decision boundary is given in Figure 11.

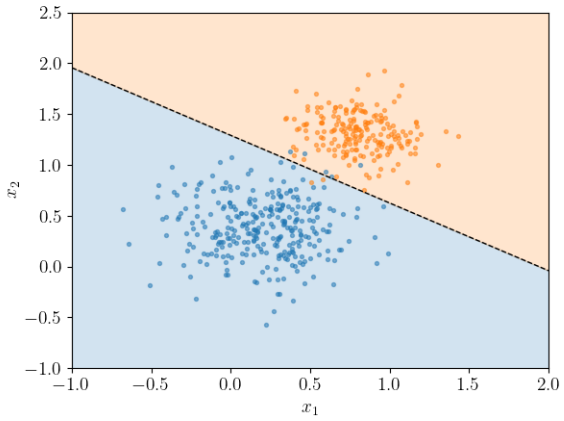

Fig. 11. Linear decision boundary of logistic regression

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

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

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

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

$$\hat{y} = \begin{cases} C1, & \text{if } \hat{p} < \text{Threshold} \\ C2, & \text{otherwise} \end{cases}$$

where $C1$ and $C2$ are the two classes. The threshold is usually set to 0.5.

Parameter Estimation

The parameters can be estimated using Maximum Likelihood Estimation (MLE). Let the number of data points be M and i be an integer such that $1 \leq i \leq M$; i is used to index a datapoint in the dataset. Let y_i denote the ground truth corresponding to datapoint x_i. Then the Likelihood, given by $L(\mathbf{w})$ is:

$$\begin{aligned} L(\mathbf{w}) &= P(y_1, y_2, ..., y_m | x_1, x_2, ..., x_m) \\ &= \Pi_{i=1}^{M} P(Y_i = y_i | X_i x_i, \mathbf{w}) \\ &= \Pi_{i=1}^{M} \sigma(y_i \mathbf{w^T} x_i) \\ \log L(\mathbf{w}) &= \sum_{i=1}^{M} \log \sigma(y_i \mathbf{w^T} x_i) \end{aligned}$$

We could maximise log-likelihood or minimise negative log-likelihood to estimate the parameters. Or equivalently, minimise Empirical Logistic Loss function, $\hat{R}(\mathbf{w})$ defined as:

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

The goal is to minimize $\hat{R}(\mathbf{w})$; however there exists no closed form solution. So we make use of Gradient Descent. The steps involved in the gradient descent algorithm for logistic regression are briefly outlined below:

- Initialize $\mathbf{w_t} = \mathbf{w_0}$ randomly.
- Repeat till convergence:

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

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

Regularised Logistic Regression

In order for the model to generalise well and to prevent over-fitting, a penalty function could be added to the loss function. We choose the $L2$ norm and hence this is termed Ridge regression. For the $L2$ norm penalty function, the empirical loss function becomes:

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

where λ denotes the regularization parameter, which is treated as a hyper-parameter. The modified expression of $\hat{R}(\mathbf{w})$ is then used for performing gradient descent.

IV. Modelling

In this section, we discuss the application of the logistic regression model to our problem.

The logistic regression is trained on the training set using gradient descent. The training process is abstracted out through the sklearn library. The only **hyper-parameter present in the model is λ,** which is the **regularization parameter.** The **best value of λ is found out using cross-validation,** which involves breaking up the training data into multiple sets, evaluating different values of λ and choosing the most suitable value.

The predictive model for a classification problem can be evaluated using multiple metrics. A short description of the most commonly used metrics is given below:

- **Accuracy:** Accuracy is simply the **ratio of number of correct predictions to total number of predictions.** Although this seems like a very good metric intuitively, accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on datasets with an equal class distribution.
- **Precision:** Precision is the **ratio of true positives to the total positive predictions.** Precision is typically used when the cost of false positive is high. For instance, email spam detection.
- **Recall:** Precision is the **ratio of true positives to the total positive ground truths.** Recall is typically used when the cost of false negative is high. For instance, in fraud detection or sick patient detection.

TABLE I
Evaluation of Logistic Regression model

Metric	Score
Accuracy	0.831
Precision	0.791
Recall	0.759
F1 Score	0.775

- **F1-score:** F1-score is simply a **harmonic average of precision and recall.** F1 Score is typically used if we need to seek a balance between Precision and Recall and there is an uneven class distribution.

The confusion matrix for the training set is shown in Figure 12. The values for the evaluation metrics are given in Table I. From this analysis, it is clear that we can **predict the survival chance of an individual to a reasonable extent from the parameters used.** This indicates that there indeed exists some correlation between survival chance and factors like socioeconomic status and gender. Predictions were also made for the entries in the test set. However, the quality of predictions could not be evaluated as labels were not available.

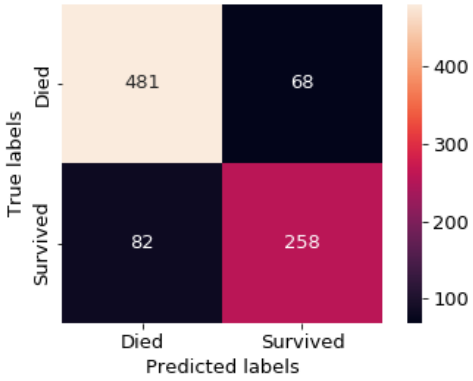

Fig. 12. Confusion matrix for the training set

ROC curve is another common tool used with binary classifiers. A Receiver Operating Characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The ROC curve for the logistic regression model is given in Figure 13.

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 0.876.**

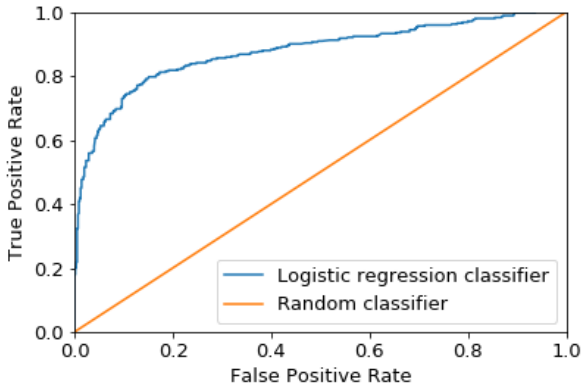

Fig. 13. ROC curve

TABLE II
EVALUATION OF KERNEL-BASED LOGISTIC REGRESSION MODEL

Metric	Score
Accuracy	0.901
Precision	0.854
Recall	0.894
F1 Score	0.873

Kernel-based logistic regression

Although the vanilla logistic regression model works reasonably well, it still suffers from that logistic regression assumes linearity in the conditional probability space. To capture non-linear relationships, it is necessary to incorporate kernels into the vanilla logistic regression model. We approximate a Gaussian kernel using Nystroem approximation.

The Nystroem method is a general method for low-rank approximations of kernels. It achieves this by essentially subsampling the data on which the kernel is evaluated. Usually, Nystroem uses the RBF/Gaussian kernel, but it can use any kernel function or a pre-computed kernel matrix. The number of samples used, which is also th,e dimensionality of the features computed, is given by a parameter.

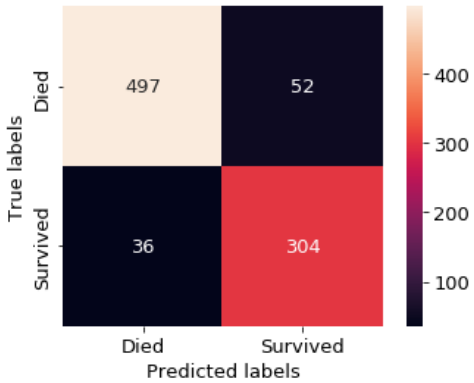

Fig. 14. Confusion matrix for the kernel-based model

The model built using this method yields the confusion matrix shown in Figure 14. The evaluation metrics for this model is presented in Table II. It can be seen that this model **clearly outperforms the vanilla logistic regression model.** This can be attributed to the use of non-linearity in the model. The ROC curve is also given in Figure 15.

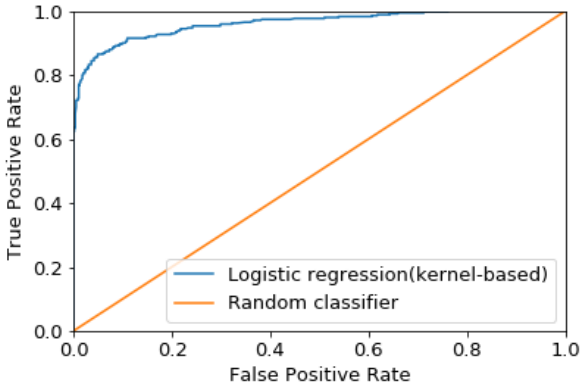

Fig. 15. ROC curve for kernel-based model

Thus, by making use of a kernel approximation, we were able to drastically enhance the performance of the logistic regression model.

V. CONCLUSIONS

From the logistic regression model, we were able to arrive at the conclusion that although luck did play a role in determining who survived and who did not, it also depended on a variety of other factors. In particular, it was found that **passengers in first class, women, children and people who paid a higher fare were all more likely to survive.** We were also able to make predictions of whether an individual will survive or not for cases where the ground truth was unknown. This analysis helps bring out biases which society as a whole may voluntarily or involuntarily hold against sections of the population. Closely reflecting on such analyses and focusing on their implications can help better everyone's lives.

VI. AVENUES FOR FURTHER RESEARCH

Although logistic regression is a powerful model, it is only a linear model. Making use of non-linear models, either by considering non-linear models or by using kernels for introducing non-linearity is an avenue worth exploring. Collecting more features (like proximity to lifeboats) could also help improve modelling.

REFERENCES

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
[4] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.
[5] “Logistic regression.” https://en.wikipedia.org/wiki/Logistic_regression.

A mathematical essay on naive bayes classifier

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this assignment is to explore the mathematical formalism behind naive bayes classifier and then to use it in a real-life application. In this assignment, as a real-life application, naive bayes classifier is used to formally identify the factors which could have been used to predict the income category of an adult, using the 1994 US census data. Data visualization, cleaning and modelling is done using Python. The analysis enables us to arrive at the conclusion that it is possible to make reasonable predictions regarding the income of an adult using factors including but not limited to education, working class and gender. This is a reworked version of the original assignment with improvements to the Modelling section in the form of enhanced benchmarking of the model including a comparison with an SVM model.**

***Index Terms*—naive bayes, python, visualization, predictive modelling, binary classification**

I. Introduction

Given a set of features and a target variable, predictive modelling is typically used for generating a model which can make predictions for cases where we do not know the value of the target variable, i.e., only the features are available. Apart from this use, a model can also be used for developing an intuition of how various factors influence the target variable. In this assignment, we try to make use of a model for the purpose of identifying the key factors which influence the decision in a classification problem.

In particular, we make use of Naive Bayes classifier for the purpose of identifying relationships in a classification problem. Naive bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. Though they are by nature very simple, they can give good results in many problem settings. Naive bayes is a very flexible classifier in the sense that it can accommodate both continuous and categorical variables. We will make use of this flexibility for designing a robust model which can solve a classification problem.

In our problem setting, the goal is to use naive bayes classifier to predict the income category of an individual given a variety of factors like education, working class, gender etc. We make use of the publicly available 1994 US census data for building the model. After building the model, we evaluate the model using a variety of evaluation metrics. By examining how well the model performs, we can identify how good the identified relationships are.

Section II gives an overview of the various techniques used for data cleaning, data visualization and an initial exploratory analysis. A lot of insights can be gained just by making qualitative observations from the given data. Section III gives a short description of the mathematical formalism behind naive bayes. Section IV describes the various models that were tried and the results that were obtained by applying naive bayes in this particular case. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we describe the process of data cleaning and data visualization. We also make some qualitative observations.

A. Preliminary analysis

The given dataset has 32561 rows and 15 columns. It must be noted that the dataset itself lacked any column headings: they had to be added in manually. A brief overview of the dataset is presented in Figure 1. We observe that we have both categorical as well as numerical variables. It also seems that there are no null values. From the common statistics (mean, median, mode) of the numerical variables, we make the following observations:

- The distribution has representatives in the age group 17 - 90. This ensures that we capture the variation across different age groups.
- Every individual has had at least 1 year of education. Note that this might not have been the case had the census been done in a developing country instead of the US.
- Capital gain and loss values are 0 each for a majority of the population, since the median is 0. This could be because a large section of the population is not involved in capital asset transactions, which is rather intuitive.

We also look at the distribution for our target class, which is income. On plotting, we obtain Figure 2. We have two categories: income $\leq$ 50K (low income category) and income $>$ 50K (high income category). Of course for income, low and high may be subjective, but for our purposes we go ahead with this nomenclature. The count of individuals in the high income category is only about a third of the count of individuals in the low income category. We will have to account for this during model building.

B. Feature by feature analysis

For each feature, we perform the following analyses:

- For continuous features, we first explore how the feature itself is distributed. For Gaussian Naive Bayes to be

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  native-country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Fig. 1. Summary of the dataset

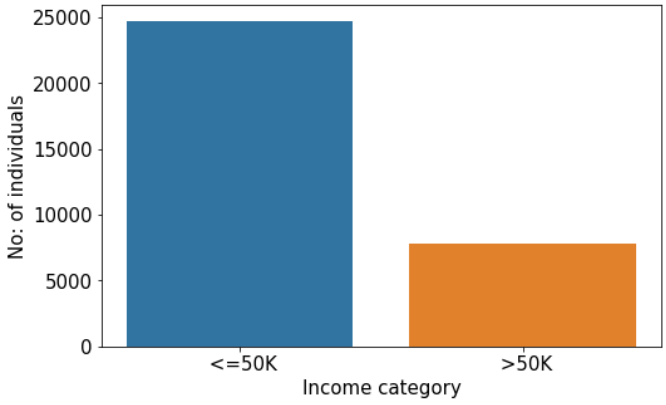

Fig. 2. Target class imbalance

effective an important requirement is to have the variable distributed approximately normally.

- Then we make appropriate plots for continuous and categorical features to explore if we can derive any insights from a visual standpoint. Only features which provide insights are reported though all features were tested.

Continuous variables

We make use of kernel density estimate (kde) plots for examining the distribution of continuous variables. KDE plots are essentially smoothed histograms. Since they are less cluttered versions of histogram plots, insights are more obvious.

For applying naive bayes, it is essential to ensure that none of the continuous features are correlated. The correlation matrix is shown in Figure 3. **Clearly, the correlation values are very small, thereby validating the naive bayes assumption of independence of features.**

We explore the continuous variables one by one below:

Age

We obtain the plot as given in Figure 4. Although the plot does resemble a Gaussian, it is skewed. In an attempt

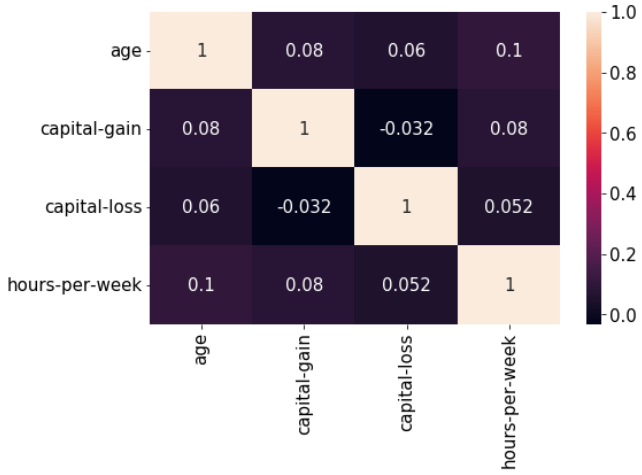

Fig. 3. Correlation for continuous features

to make it more like a Gaussian, we can apply the box-cox transformation. It transforms a datapoint y as follows:

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0 \\ \log y, & \text{if } \lambda = 0 \end{cases}$$

λ is treated as a hyper-parameter and the most suitable value is chosen. After performing the box-cox transformation, the age variable distribution gets transformed and the transformed feature is illustrated in Figure 5. Clearly, the distribution more closely resembles a Gaussian now. We will use this transformed quantity while applying Gaussian Naive Bayes.

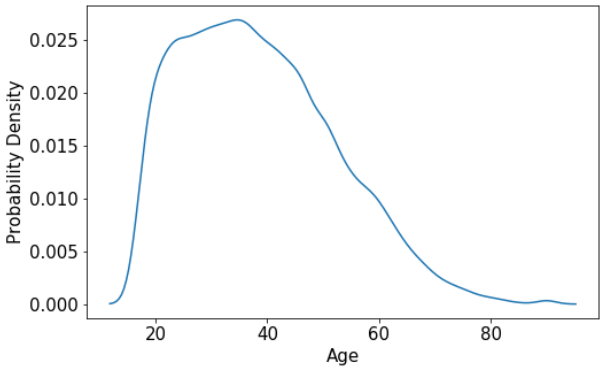

Fig. 4. Density plot for Age

On plotting the distribution of age according to income category, we obtain the plot in Figure 6. We observe that:

- The **peak for income group $\leq$ 50K occurs close to the age 20.** This is intuitive as we expect people in their 20s to make less income as they are probably just starting out on jobs/ businesses.
- **The peak for income $>$ 50K occurs close to the age 40.** This is also intuitive as we expect most people to be at their peak earning capacity around this age. As they grow

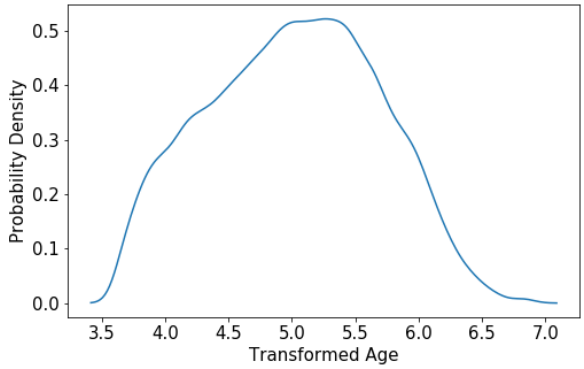

Fig. 5. Density plot for transformed age variable

older, they move closer to retirement, possibly adversely affecting their income.

- We notice that there is a **greater proportion of people with income $\leq$ 50K as compared to $>$ 50K for any given age,** which is what we would expect from any society.

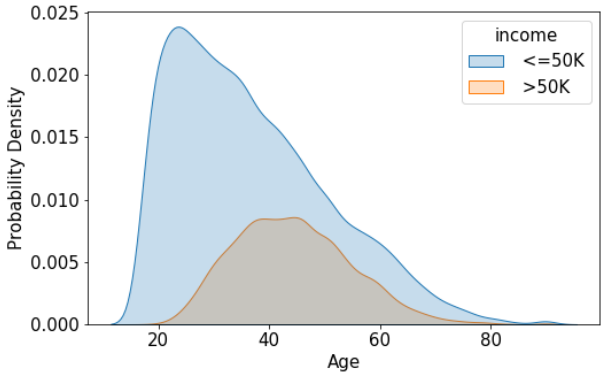

Fig. 6. Age distribution for different income categories

fnlwgt

No description of this variable was given in the problem statement. After scouring the internet to find what this variable actually represents, the following description was found in the official dataset description:

"People with similar demographic characteristics should have similar weights. There is one important caveat to remember about this statement. That is that since the CPS sample is actually a collection of 51 state samples, each with its own probability of selection, the statement only applies within state."

However, we clearly have no information regarding which state an individual is from. **Hence, for our purposes, we can safely discard this variable** as this variable does not add any value, as long as we don't have any information regarding state. For the sake of completeness, the distribution of this variable is given in Figure 7.

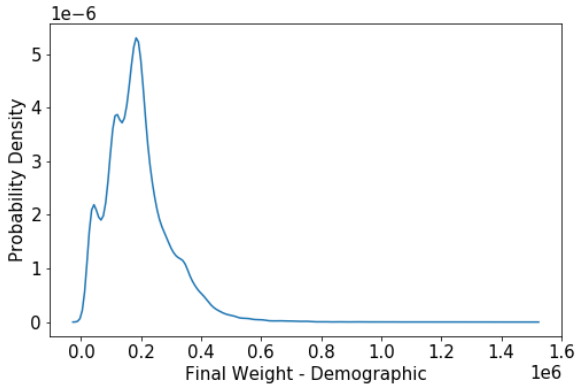

Fig. 7. Distribution of the variable fnlwgt

Years of education

We obtain the distribution plot as given in Figure 8. We see that the distribution doesn't resemble a Gaussian at all. We also note that whatever information is captured by years of education is also captured by the 'education' column, which is categorical and hence is easier to handle with naive bayes. So we drop this column and do not use it for Naive Bayes model.

On plotting the distribution of years of education according to income category, we obtain the plot in Figure 9. We observe that in general, **people who earn more tend to have invested a longer duration in education.**

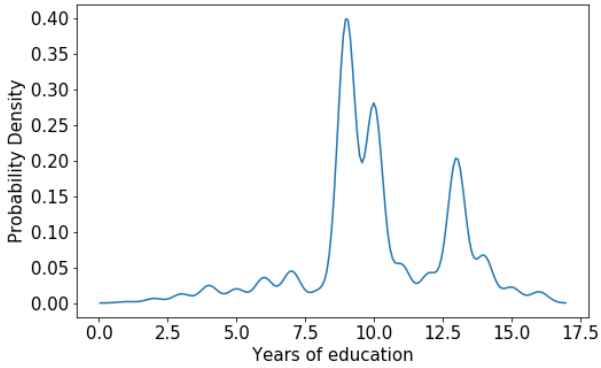

Fig. 8. Distribution of the number of years of education

Capital gain

We obtain the distribution plot as given in Figure 10. Although the right tail is longer than left tail, since the values are almost zero along the right tail, we consider this to be a Gaussian and hence do not apply any transform.

On plotting the distribution of capital gain according to income category, we obtain the plot in Figure 11. Capital

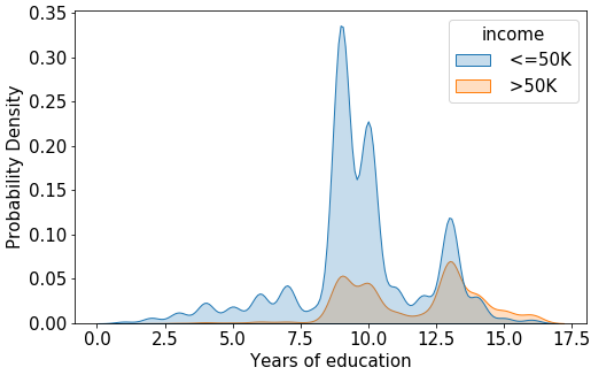

Fig. 9. Number of years of education for different income categories

gain plots don't seem to reveal any information regarding the dataset. This could be because more than 50% of the points have the capital gain value as 0 as a majority of the population do not involve in capital asset transactions. So we try plotting only the non-zero values to see if we can infer anything. We obtain the plot given in Figure 12. Clearly, we observe that a **higher capital gain has a higher probability density for people in the high income bracket,** which makes sense intuitively.

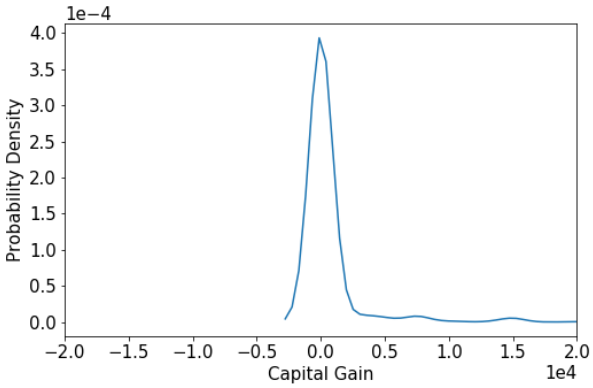

Fig. 10. Distribution of the capital gain

Capital loss

We obtain the distribution plot as given in Figure 13. Again the right tail is longer than left tail, however since the values are almost zero along the right tail, we consider this to be a Gaussian and hence do not apply any transform.

Since the situation is same as that of capital gain, so we try plotting only the non-zero values to see if we can infer anything. We obtain the plot given in Figure 14. Counter-intuitively, it seems like people with higher income have higher capital losses on average! However, this could be because the more you get involved in capital asset transactions, the more are your chances for gain/loss. Individuals could still be in the high income category if the gains offset losses.

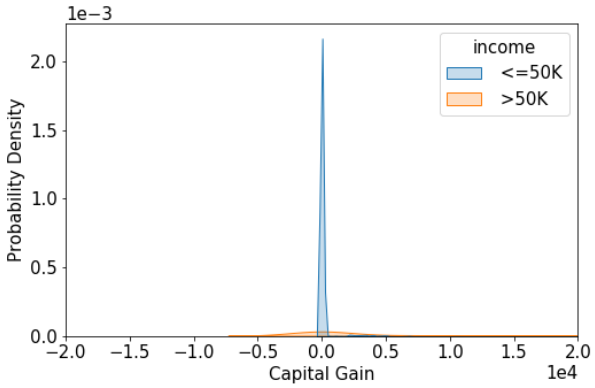

Fig. 11. Capital gain for different income categories

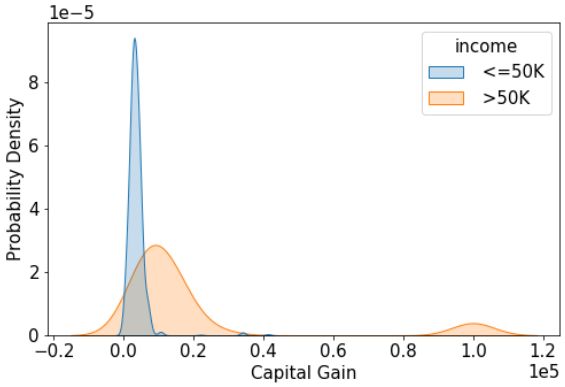

Fig. 12. Capital gain considering only non-zero values

Hours per week

We obtain the distribution plot as given in Figure 15. The distribution is sufficiently close to Gaussian, so we do not use any transforms.

On plotting the distribution of working hours per week according to income category, we obtain the plot in Figure 16. We make the following observations:

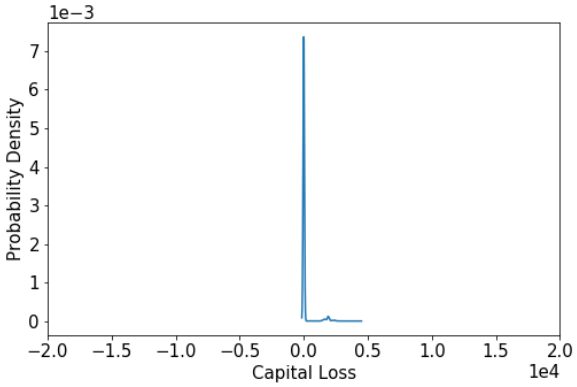

Fig. 13. Distribution of the capital loss

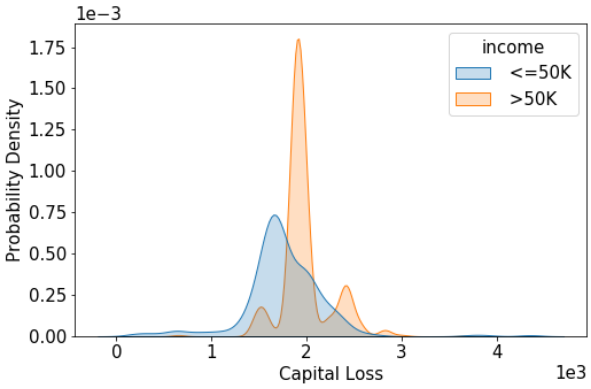

Fig. 14. Capital loss considering only non-zero values

- Contrary to what one would expect, it appears that working more hours does not necessarily mean your income will be higher.
- However, it can be seen that a higher proportion of high income population works for more than 40 hours as compared to the lower income population.

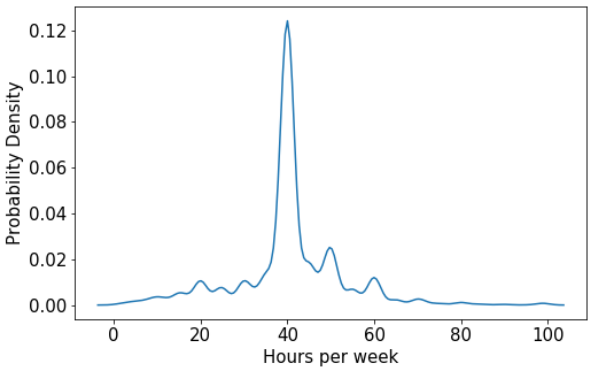

Fig. 15. Distribution of the working hours per week

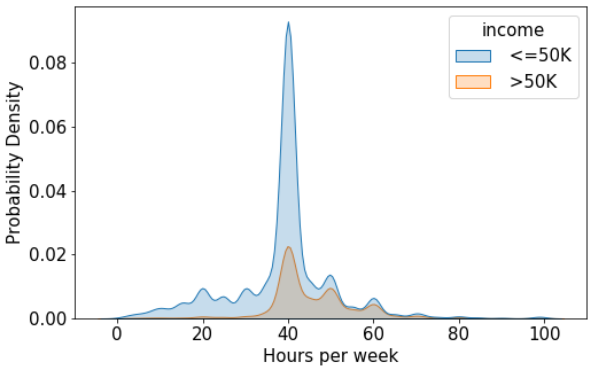

Fig. 16. Working hours per week for different income categories

Categorical variables

For the case of categorical variables, we make use of count plots instead of kde plots for visualization. We plot the count of each category of a categorical variable according to the two income categories.

Working class

We obtain the count plot as shown in Figure 17. We make the following observations:

- Self-employed(inc) is the only category with more individuals belonging to high income category as compared to low income category. This shows that you are more **likely to earn better if you manage to start a successful business for yourself.**
- Private sector has a high proportion of people from both high-income and low-income categories. This evidences the infamous **income disparity in the private sector.**
- There are some individuals for which the working class is denoted by ?'. These represent individuals for who we do not know the working class.

On examination, it is found that only 5.6% of the total number of entries have the working class missing. Since the number is marginal, we drop these rows.

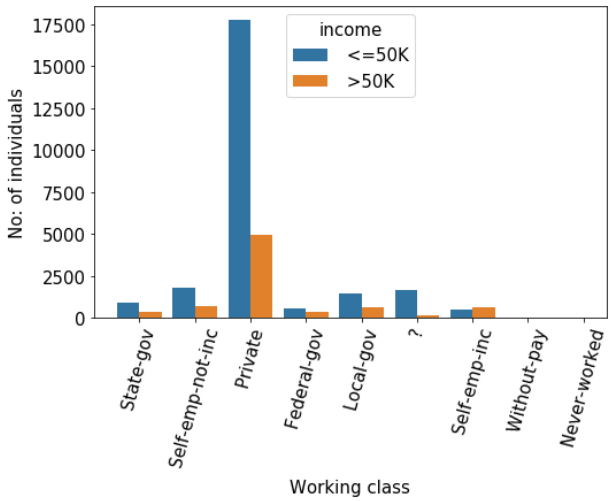

Fig. 17. Distribution of working class

Education

We obtain the count plot as shown in Figure 18. We make the following observations:

- Although we cannot make a blanket statement that education directly ensures high income, we see that the **only categories with proportion of high income group more than proportion of low income group are Masters and Doctorate groups.**
- The proportion of high income individuals is very low for people who did not attend university, again implying that **education has a significant role in determining income.**

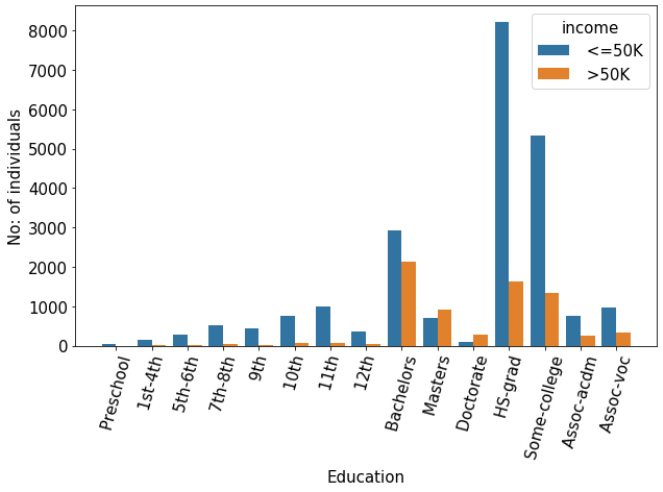

Fig. 18. Distribution of education levels

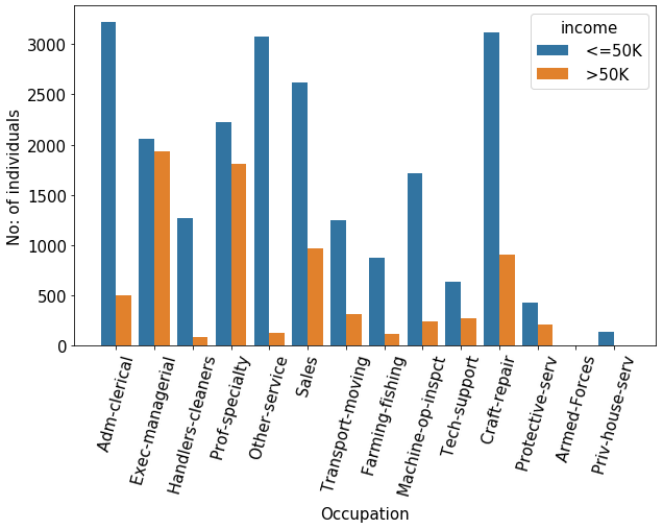

Fig. 20. Distribution of occupation

Marital status

We obtain the count plot as shown in Figure 19. We observe that Married-civ-spouse is the only category which has comparable number of people belonging to both categories of income. For other categories, the fraction of people belonging to high income category is marginal. This might be because married individuals may be more concerned about their financial security.

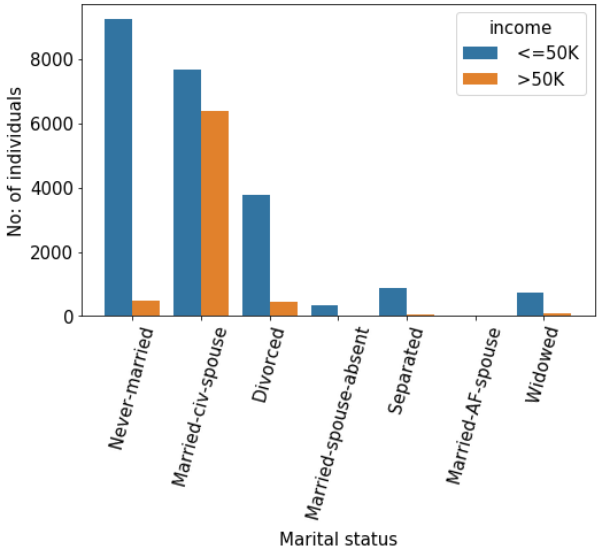

Fig. 19. Distribution of marital status

Occupation

We obtain the count plot as shown in Figure 20. We observe that **managerial executives and speciality professionals are likely to earn more.** This also makes sense intuitively as these are often considered as the "high-paying" professions.

Gender

We obtain the count plot as shown in Figure 21. We observe that **on an average, males have more chance to earn more than females,** and that about a third of males earn more than 50k income.

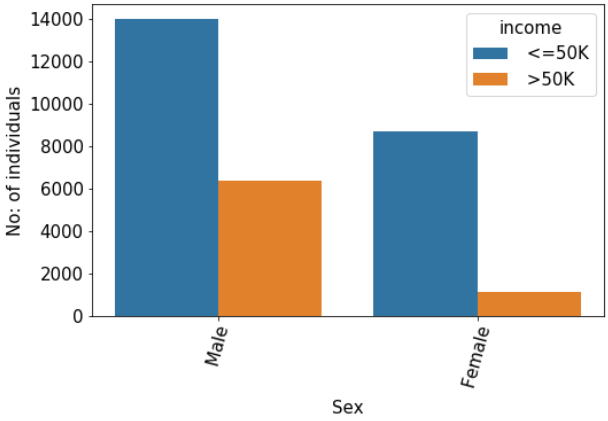

Fig. 21. Distribution of gender/sex

III. Model: Naive Bayes Classifier

In this section, we will give a brief overview of the mathematical formalism behind the naive bayes classifier.

Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with **strong (naive) independence assumptions between the features.**Naive Bayes classifiers are **highly scalable**, requiring a number of parameters linear in the number of variables/features in a learning problem. Maximum-likelihood training, which is the most generally used strategy, can be done by **evaluating a closed-form expression** which takes linear time, rather than by expensive iterative approximation, such as gradient descent, as used for many other types of

classifiers. Despite the seemingly strong assumptions, naive bayes classifiers have been shown to work well in a lot of real-life scenarios.

A. Probabilistic Model

In any classification problem, our goal is to find the class C_k in which a given datapoint **x**, which will be an n-dimensional vector, where n is the number of features, belongs. Then, more formally, our goal is to find $P(C_k|\mathbf{x})$ for each class C_k.

Using Bayes' theorem,

$$P(C_k|\mathbf{x}) = \frac{P(C_k, \mathbf{x})}{P(\mathbf{x})}$$

For a given datapoint, the denominator is fixed, implying that the class to which the point should be assigned is influenced solely by the numerator which essentially contains the joint probability function $P(C_k, \mathbf{x})$. Now,

$$P(C_k, \mathbf{x}) = P(C_k, x_1, x_2, ..., x_n)$$

where $x_1, x_2..., x_n$ are the n features. **This is where we make use of the strong naive bayes assumption of mutual independence of features, conditional on the class** C_k Using this assumption,

$$P(C_k, x_1, x_2, ..., x_n) = P(C_k)P(x_1|C_k)P(x_2|C_k)...P(x_n|C_k)$$

Therefore, we can write:

$$P(C_k|\mathbf{x}) \propto P(C_k)\Pi_{i=1}^{n}P(x_i|C_k)$$

$P(C_k)$ is termed the **prior probability** for class C_k, since this can be interpreted as the probability of observing class C_k before any observations were made. If we know the distribution of the classes before measurements through some **domain knowledge, this can be used for assigning the prior probabilities.** If not, we can always choose a uniform prior which would mean assigning equal probabilities to all classes.

By making the simplifying assumption of independence of features, we are required to compute only the distributions $P(x_i|C_k)$ for $i \in \{1, 2, ..., n\}$, instead of the joint distribution. How does this make the computations easier?

To see this, let us assume all distributions are normal. If we do not assume conditional independence, we will be required to compute $P(x_1, x_2, ..., x_n|C_k)$ which essentially is a joint Gaussian having n variables. Clearly, the covariance matrix of such a Gaussian will n^2 entries and hence the estimation of parameters will be $O(n^2)$. However, if we assume independence, this can be broken down into n separate Gaussians with just 1 variable each. Clearly, such a model would only require $O(n)$ parameter estimations, giving us a **quadratic speedup**. This can immensely help in computations, especially while dealing with reasonably large datasets.

Notice that this simplifying assumption reduces the complexity of the model. This makes naive bayes a **high bias, low variance classifier.**

Typically, for continuous features we assume the distribution to be Gaussian and for categorical variables, we assume the distribution to be bernoulli/categorical distribution. After choosing a distribution, we estimate the parameters using Maximum Likelihood Estimation.

B. MAP decision rule

After obtaining the probabilities $P(C_k|\mathbf{x})$, we must make a prediction to assign a class. The most commonly used prediction strategy is to **pick the class with the highest value** of $P(C_k|\mathbf{x})$. This is termed as the maximum a posteriori or MAP decision rule.

C. Gaussian Naive Bayes

Gaussian Naive Bayes is a typically used model for continuous variables. We essentially assume the Gaussian distribution:

$$f_x(x|C_k) = \frac{1}{\sqrt{2\pi}\sigma_k}e^{\frac{-(x-\mu_k)^2}{2\sigma_k^2}}$$

for the probability density function for class C_k.

We estimate μ_k, σ_k and the prior $\pi_k(P(C_k))$for each class C_k as follows:

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

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

$$\hat{\pi}_k = \frac{n_k}{n}$$

where n_k is the number of observations of the k$^{\text{th}}$ class and n is the total number of observations.

If we assume shared variances across the different classes, we will obtain **linear decision boundaries.**

D. Categorical Naive Bayes

When the input feature is categorically distributed, we use categorical naive bayes. In categorical naive bayes, we assume the probability distribution for each feature to be a categorical variable. Suppose a featurex_j can take values in categories $1, 2, ..., T$. Then mathematically, the probability for feature x_j to take value as category t, given a class C_k is estimated as:

$$P(x_j = t|C_k) = \frac{N_{kt}}{N_k}$$

where N_{kt} is the number of samples in class C_k which have category of x_j as t and N_k is the total number of samples in class C_k. We then calculate the probability of observing the sample X_i given a class C_k by multiplying out the above calculated probabilities for each feature.

However, this approach has one drawback: if we encounter a category which we have never seen before, the entire product vanishes because the probability for this particular category goes to zero. To prevent this, a smoothing parameter α, which is typically a small value greater than zero, is added to both the numerator and denominator:

TABLE I
METRICS FOR MODEL 1 (UNIFORM PRIOR)

Metric	Score
Accuracy	0.831
Precision	0.692
Recall	0.578
F1 Score	0.630

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

IV. MODELLING

In this section, we discuss the application of the naive bayes classifier to our problem.

We create and evaluate two separate models:

1) In Model 1, we **set the prior uniformly for both classes,** i.e., we assign 0.5 each for the prior values.
2) In Model 2, we **set the prior according to the class distribution,** i.e we set 0.751 for the low-income category and 0.249 for the high-income category.

Each model is generated using a **"mixed" naive bayes approach.** This is because we have both numerical and categorical variables. So we fit a Gaussian naive bayes classifier on the continuous features, a categorical naive bayes classifier on the categorical features, get the output probabilities and multiply them out to get the total probabilities from the model.

We evaluate the models based on multiple metrics. A short description of the used metrics is given below:

- **Accuracy:** Accuracy is simply the **ratio of number of correct predictions to total number of predictions.** Although this seems like a very good metric intuitively, accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on datasets with an equal class distribution.
- **Precision:** Precision is the **ratio of true positives to the total positive predictions.** Precision is typically used when the cost of false positive is high. For instance, email spam detection.
- **Recall:** Precision is the **ratio of true positives to the total positive ground truths.** Recall is typically used when the cost of false negative is high. For instance, in fraud detection or sick patient detection.
- **F1-score:** F1-score is simply a **harmonic average of precision and recall.** F1 Score is typically used if we need to seek a balance between Precision and Recall and there is an uneven class distribution.

A. Model 1

The confusion matrix for model 1 is shown in Figure 22. The values for the evaluation metrics are given in Table I.

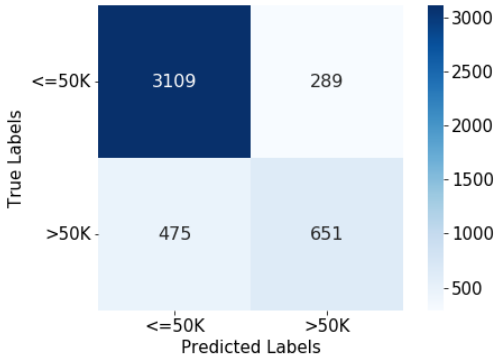

Fig. 22. Confusion matrix for model 1 (uniform prior)

TABLE II
METRICS FOR MODEL 2 (NON-UNIFORM PRIOR)

Metric	Score
Accuracy	0.811
Precision	0.746
Recall	0.363
F1 Score	0.489

B. Model 2

The confusion matrix for model 2, using prior based on class distribution, is shown in Figure 23. The values for the evaluation metrics are given in Table II.

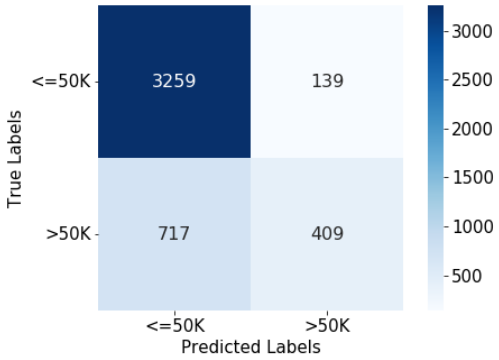

Fig. 23. Confusion matrix for model 2 (non-uniform prior)

We observe that **Model 2 has better precision whereas Model 1 performs better in all other metrics.** Although we might consider Model 2 in settings where precision is of utmost importance, in this case since Model 1 does better in all other aspects, we choose **Model 1 to be the better model.**

ROC curve is another common tool used with binary classifiers. A Receiver Operating Characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various

TABLE III
METRICS FOR GAUSSIAN NAIVE BAYES

Metric	Score
Accuracy	0.787
Precision	0.650
Recall	0.313
F1 Score	0.423

threshold settings. The ROC curve for Model 1 is given in Figure 24.

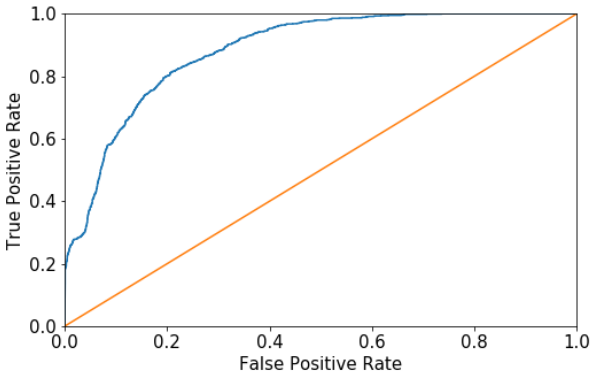

Fig. 24. ROC curve for Model 1

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 0.882.**

To get a sense of how much continuous variables and categorical variables contribute to the prediction quality, we take the Gaussian naive bayes and categorical naive bayes (with uniform prior) separately and make predictions. For Gaussian naive bayes, we obtain the confusion matrix shown in Figure 25 and metrics shown in Table III. For categorical naive bayes, we obtain the confusion matrix shown in Figure 26 and metrics shown in Table IV.

From the confusion matrix as well as the metrics, it is clear that the Gaussian naive bayes contributes mostly to the precision metric whereas the categorical naive bayes dominates all other metrics. If we were to choose between the two, **categorical naive bayes is more relevant as it leads to a more balanced model. Hence, we can conclude that the categorical variables contribute more to the income category prediction.**

Comparison with SVM

Despite trying multiple configurations for the naive bayes model, the model still has rather low values for the evaluation metrics. This is probably because of the underlying naive

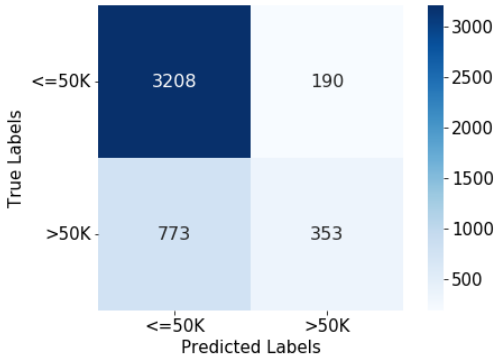

Fig. 25. Confusion matrix for Gaussian naive bayes

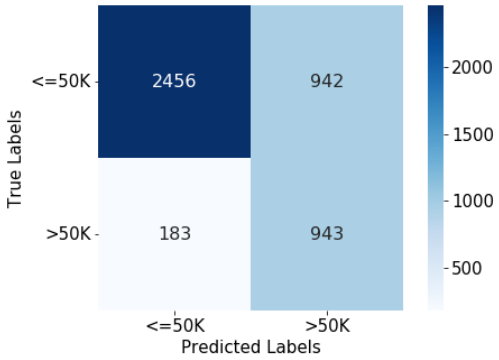

Fig. 26. Confusion matrix for categorical naive bayes

assumptions of the model. To see if this is indeed the case, we build a support vector machine (SVM) model on the same dataset. Since SVM does not have naive independence assumptions, an enhancement of performance would imply that it was indeed these assumptions which hampered the performance of the naive bayes model.

We create multiple SVM models based on values of different hyperparameters. We use the gaussian kernel and grid search on the hyperparameter space. The best model which we obtain produces a confusion matrix as shown in Figure 27. The evaluation metrics are given in Table V.

It is evident that the SVM model outperforms the naive bayes model in terms of F1-score, which is our metric of choice. It is to be noted that the SVC classifier takes much longer on average to train as compared to naive bayes. This

TABLE IV
METRICS FOR CATEGORICAL NAIVE BAYES

Metric	Score
Accuracy	0.751
Precision	0.500
Recall	0.837
F1 Score	0.626

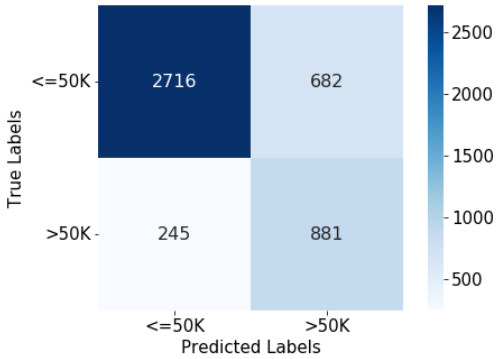

Fig. 27. Confusion matrix for SVM

TABLE V
METRICS FOR SVM

Metric	Score
Accuracy	0.795
Precision	0.564
Recall	0.782
F1 Score	0.655

is to conclude that even though SVM performs better, naive bayes does possess discriminatory characteristics to separate out the classes and hence would be useful in setting a minimum benchmark in large datasets.

V. CONCLUSIONS

From our extensive analysis of the given dataset using naive bayes calssifier, we arrive at the following conclusions:

- Naive bayes classifier, despite having rather strong assumptions, performs reasonably well as long as the features are more or less independent and the continuous variables are approximately Gaussian.
- Changing the prior can significantly affect the performance of the naive bayes model as evidenced by the improvement in performance using a uniform prior instead of class weighted prior.
- Categorical features provide more insights regarding the value of target variable as compared to continuous features as is evident from the fact that the categorical naive bayes classifier outperforms the Gaussian naive bayes classifier.
- The income bracket of an individual can be predicted to a reasonable extent using factors like education, age, profession and gender.
- While it is acceptable that education and working hours influence the income, it must be critically examined if certain groups based on gender and race face barriers while aspiring to be in the high income category to ensure wellness and propserity of the society as a whole.

VI. AVENUES FOR FURTHER RESEARCH

Naive bayes classifier, despite working reasonably well, has very strong assumptions which might not be warranted in many cases. Using other models like SVM which do not require such strong assumptions might lead to better results. Further, rather than using Gaussian naive bayes, if we can more accurately model the distribution of continuous variables, we might be able to achieve better performance.

REFERENCES

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[5] "Naive bayes classifier." https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

A mathematical essay on decision tree

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this assignment is to explore the mathematical formalism behind decision tree classifier and then to use it in a real-life application. In this assignment, as a real-life application, decision tree classifier is used to formally identify the factors which could have been used to predict how acceptable a car is from the popular car evaluation dataset. Data visualization, cleaning and modelling is done using Python. The analysis enables us to arrive at the conclusion that it is possible to make reasonable predictions regarding the acceptability of a car using factors including but not limited to safety rating, price and luggage boot space. This is a reworked version of the original assignment with improvements to the Modelling section in the form of enhanced benchmarking of the model including a comparison with an XGBoost model. The plots have also been improved to enhance the clarity of presentation.**

***Index Terms*—decision tree, python, visualization, predictive modelling, multinomial classification**

I. Introduction

Given a set of features and a target variable, predictive modelling is typically used for generating a model which can make predictions for cases where we do not know the value of the target variable, i.e., only the features are available. Apart from this use, a model can also be used for developing an intuition of how various factors influence the target variable. In this assignment, we try to make use of a model for the purpose of identifying the key factors which influence the decision in a classification problem.

In particular, we make use of decision tree classifier for the purpose of identifying relationships in a classification problem. Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.. Though they are by nature very simple, they can give good, and more importantly, interpretable results in many problem settings. Decision tree is a very flexible classifier in the sense that it can accommodate both continuous and categorical variables. In our particular problem, we will restrict ourselves to a classification problem using categorical variables.

In our problem setting, the goal is to use decision tree classifier to predict the acceptability level of a car given a variety of factors like price, number of doors, passenger capacity, luggage boot space and safety rating. We make use of the publicly available car evaluation dataset for building the model. After building the model, we evaluate the model using a variety of evaluation metrics. By examining how well the model performs, we can identify how good the identified relationships are. Also, we check how critical each feature is to the prediction and create a ranking based on the importance of features.

Section II gives an overview of the various techniques used for data cleaning, data visualization and an initial exploratory analysis. A lot of insights can be gained just by making qualitative observations from the given data. Section III gives a short description of the mathematical formalism behind decision tree. Section IV describes the various models that were tried and the results that were obtained by applying decision tree in this particular case. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we describe the process of data cleaning and data visualization. We also make some qualitative observations.

A. Preliminary analysis

The given dataset has 1728 rows and 7 columns. It must be noted that the dataset itself lacked any column headings: they had to be added in manually. A brief overview of the dataset is presented in Figure 1. We observe that we have **only categorical variables.** It also seems that there are no null values.

We also look at the distribution for our target class, which is the acceptability rating termed 'target'. On plotting, we obtain Figure 2. We have four categories: unacc (Unacceptable), acc (Acceptable), good (Good) and vgood (Very good). The count of cars in the unacceptable category is disproportionately high compared to other categories. We will have to account for this during model building.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   target    1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Fig. 1. Summary of the dataset

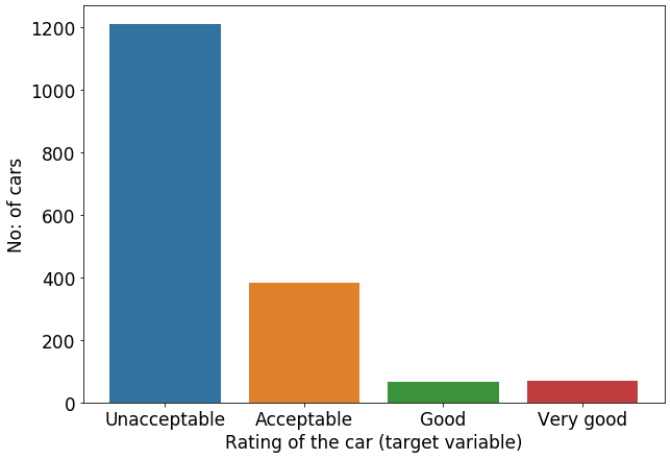

Fig. 2. Target class imbalance

B. Feature by feature analysis

Since all 6 features are categorical, for each feature we create a count plot, separating out the target classes. This can give qualitative insights regarding how a feature may affect the target variable.

Buying price

We obtain the count plot as shown in Figure 3. We make the following observations:

- Cars with high and very high buying price have a large proportion of unacceptable vehicles. Furthermore, there are no good or very good vehicles at this price range.
- Medium and low priced cars have representation from all four target categories.

Intuitively, this makes sense since people generally tend to find high prices unacceptable. For a deal to be considered good/ very good, it is almost always the case that the price should be on the lower end.

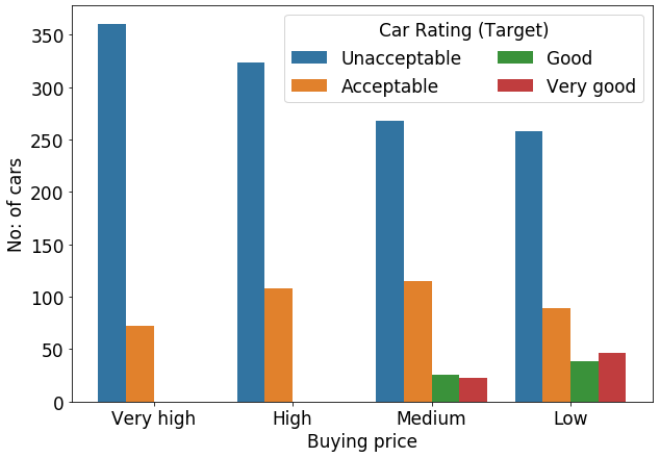

Fig. 3. Distribution of buying price

Maintenance price

We obtain the count plot as shown in Figure 4. We observe that the distribution is largely identical to the distribution for buying price. Again, we expect people to find high maintenance cost to be generally unacceptable.

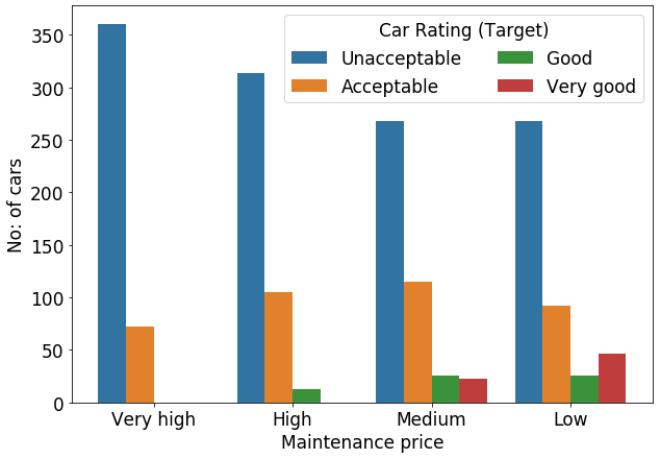

Fig. 4. Distribution of maintenance price

Number of doors

We obtain the count plot as shown in Figure 5. We observe that the distribution is largely identical for each category, i.e., irrespective of what the number of doors is, the distribution is same. In other words, the number of doors doesn't really provide any information regarding what would be the acceptability rating of a car. This implies that number of doors is not really a deciding factor as far as acceptance rating is concerned.

Passenger capacity

We obtain the count plot as shown in Figure 6. We make the following observations:

- Cars that can accommodate only two people are all unacceptable.
- Cars which can accommodate four and more than four have similar distributions.

This seems to suggest that while having at least 4 seats is essential, beyond that the number of seats doesn't really have an impact on the acceptability rating.

Size of luggage boot

We obtain the count plot as shown in Figure 7. We observe that though all categories have representation from all target categories, in general, the rating increases as the luggage boot size increases, which is what we would expect intuitively anyway.

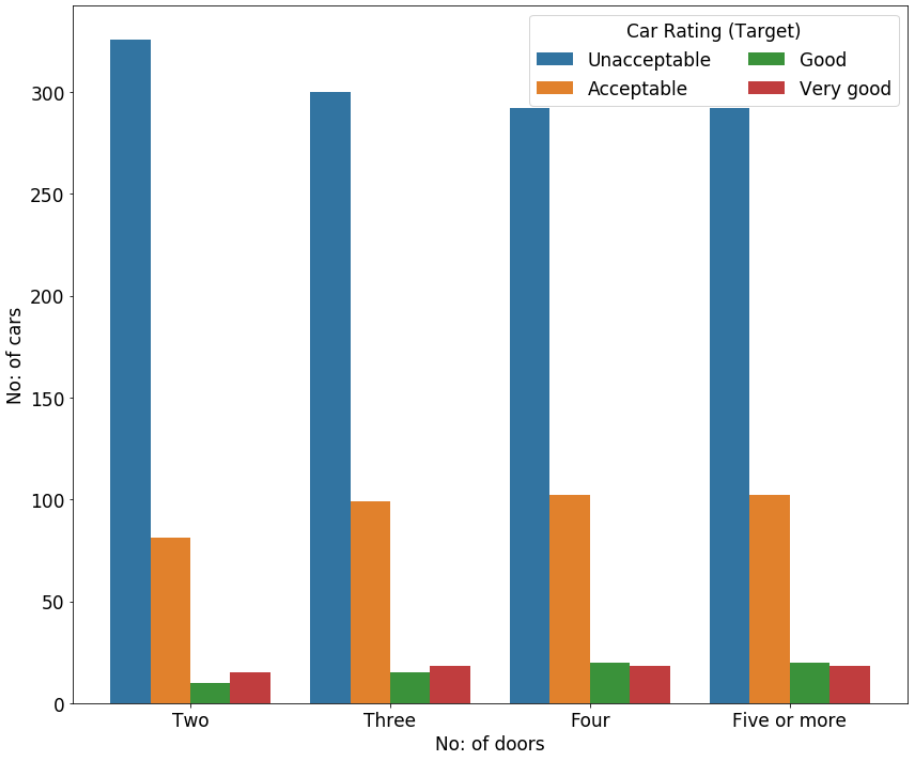

Fig. 5. Distribution of number of doors

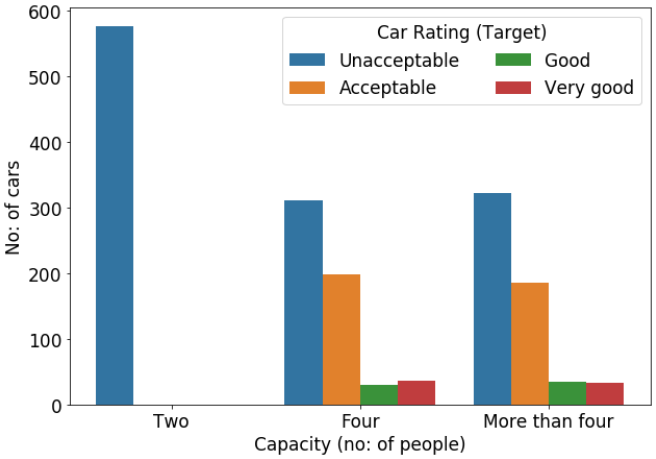

Fig. 6. Distribution of passenger capacity

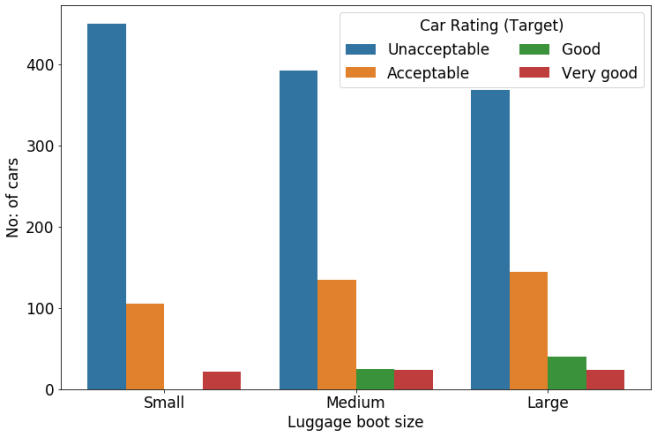

Fig. 7. Distribution of luggage boot size

Safety rating

We obtain the count plot as shown in Figure 8. We make the following observations:

- All low safety rated cars are unacceptable, which is what we would expect.
- As safety level increases, proportion of unacceptable cars decreases.

This seems to suggest that safety is a strong factor in predicting the acceptability of a car.

III. Model: Decision Tree Classifier

In this section, we will give a brief overview of the mathematical formalism behind the decision tree classifier.

In general, decision trees come under the broad umbrella of algorithms called CART algorithms: Classification and Regression Tree algorithms. Decision tree is a specific algorithm within this broad class. Decision trees themselves are of two kinds:

1) Classification trees: When the predicted outcome is the class (discrete) to which the data belongs.

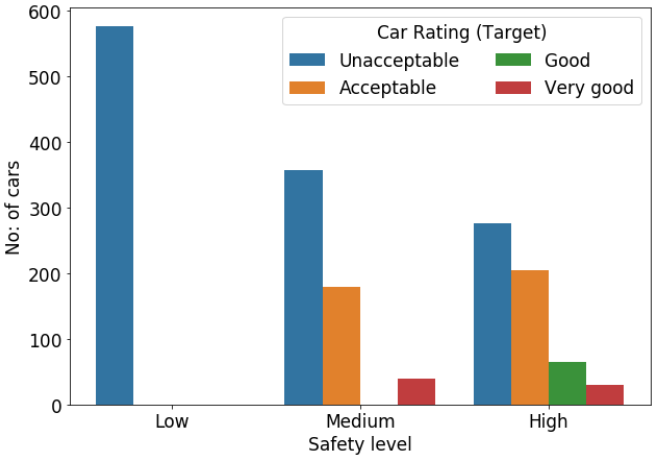

Fig. 8. Distribution of safety levels

2) Regression trees: When the predicted outcome can be considered a real number.

In this assignment, we will be making use of classification trees since the problem is a multi-label classification problem.

A. Algorithm Overview

Before describing the algorithm used by a decision tree classifier, it is necessary to make oneself familiar with the terminology and structure of a decision tree. A prototypical decision tree is shown in Figure 9. A brief overview of the terminology:

- Root node: Root node is from where the decision tree starts. It represents the entire dataset.
- Decision node: Decision node represents a node where a decision has to be made regarding a split to the dataset under current consideration.
- Leaf node: Leaf node represents the final output node. The tree cannot be segregated further after reaching a leaf node.

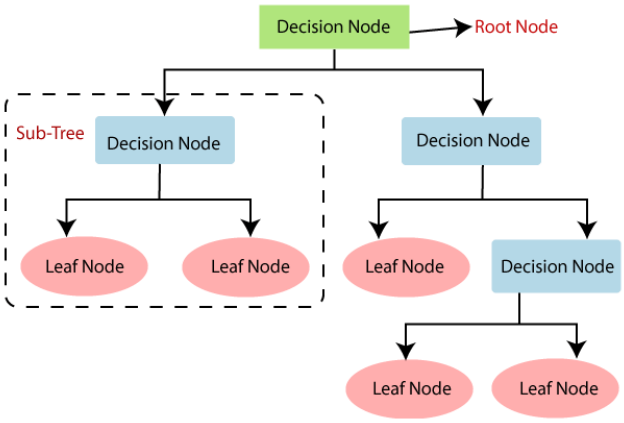

Fig. 9. Structure of a decision tree

A decision tree classifier makes used of a **rule-based classification algorithm** as opposed to a probabilistic model, which is adopted by algorithms like naive bayes. This means that rather than learning a number of parameters and then using this for predicting the probability of a sample belonging to a particular class, a set of learnt rules are used for directly predicting the class of a sample. In a sense, this rule-based approach mimics human decision making process.

The general structure of the algorithm used by a decision tree classifier is outlined below:

1) Begin the tree with the root node which contains the complete dataset.
2) Find the best attribute in the dataset using Attribute Selection Measure (ASM). More details regrading ASM will be discussed in the following section.
3) Split the dataset my making a decision rule based on an optimal value of the best attribute.
4) Recursively make new decision trees using the subsets of the dataset created in step 3. Continue this process until you reach a leaf node.

Once the tree has been built, classifying a new sample is as simple as traversing the tree till you reach a leaf node, which will provide the predicted class.

The major benefits offered by a decision tree classifier are:

- **Uses a white box/open-box model:** If a given situation is observable in a model the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model, the explanation for the results is typically difficult to understand, for example with an artificial neural network.
- **Non-parametric approach**: Makes no assumptions of the training data or prediction residuals; e.g., no distributional, independence, or constant variance assumptions.
- **Requires little data preparation**: Other techniques often require data normalization. Since trees can handle qualitative predictors, there is no need to create dummy variables

Some limitations of this approach:

- **Trees are unstable:** A small change in the training data can result in a large change in the tree and consequently the final predictions.
- **Non-optimality**: Practical decision-tree learning algorithms are based on heuristics such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.

B. Mathematical formalism

What a decision tree essentially learns is a decision for each decision node. In particular, suppose a decision tree makes use of an impurity measure S (various measures for quality will be discussed). Suppose a node splits the dataset into two subsets left and right. Then the node computes the quantity:

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

where m_{left} is the number of samples in the left subset, m_{right} is the number of samples in the right subset, S_{left} is the impurity measure for the left subset and S_{right} is the impurity measure

for the right subset. The attempt is to find the split which minimizes $J(k, t_k)$.

We can regard $J(k, t_k)$ as the 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 cut-point in order to split the data further so as to minimise the cost function. This process continues till the set stopping criterion is reached or when every training sample is perfectly classified.

It is to be noted that without the intervention of appropriate stopping criterion or other means, the tree could grow very deep, leading to over-fitting. 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.

An overview of two common attribute selection measures/ impurity measures is given below:

1) Gini Index/Impurity: This is essentially a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p_{i,k}}(1 - \hat{p_{i,k}})$$

where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node. The Gini index takes on a small value if all of the $\hat{p}_{i,k}$'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. The equation for Gini index can also be rewritten as:

$$G = 1 - \sum_{k=1}^{K} \hat{p_{i,k}^2}$$

2) Entropy: The formulation has a close resemblance with the entropy formulation in thermodynamics.

$$D = -\sum_{k=1}^{K} \hat{p_{i,k}} \log \hat{p_{i,k}}$$

where again K is the total number of classes and $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node. As was the case with Gini index, a smaller value of entropy signifies higher purity and a larger value of entropy signifies impurity.

C. Regularization techniques

As discussed above, allowing a tree to grow without stopping condition will inevitable lead to over-fitting. To mitigate this, we generally apply regularization techniques so as to increase the ability of the mode to generalize. We effectively tune the low bias - high variance model such that we decrease the variance at the cost of slightly increasing the bias. Two common regularization techniques are discussed below:

1) **Cost-complexity pruning:** Rather than considering every possible sub-tree, we consider a sequence of trees indexed by non-negative tuning parameter α. For each α, these correspond to a sub-tree $T \in T_0$ 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 $\alpha = 0$, the sub-tree T is the original tree T_0 . We can select a value for α using a validation set or cross-validation.

2) **Bagging:** In bagging, we construct B trees using B bootstrapped training sets and average the resulting predictions. Bootstrapping is a technique to obtain datapoints using random sampling with replacement. Although the constructed trees may be deep and hence suffer from high variance individually, averaging reduces variance and results in a better model. This, in fact, is a precursor to Random forests which in addition to this, deploys random features during splitting . Bagging improves prediction by lowering the variance, but at the expense of loss of interpretability.

IV. Modelling

In this section, we discuss the application of the decision tree classifier to our problem.

We create and evaluate three models:

1) In Model 1, we let the decision tree grow without any constraints.
2) In Model 2, we perform pruning on the tree to avoid overfitting.
3) In Model 3, we perform bagging to avoid overfitting.

For each model, we take care to account for the inherent imbalance in the training dataset with respect to the target class. This is done by making use of appropriate wweights during the training of the decision tree.

We evaluate the models based on multiple metrics. A short description of the used metrics is given below:

- **Accuracy:** Accuracy is simply the **ratio of number of correct predictions to total number of predictions.** Although this seems like a very good metric intuitively, accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on datasets with an equal class distribution.
- **Precision:** Precision is the **ratio of true positives to the total positive predictions.** Precision is typically used when the cost of false positive is high. For instance, email spam detection.
- **Recall:** Precision is the **ratio of true positives to the total positive ground truths.** Recall is typically used when the cost of false negative is high. For instance, in fraud detection or sick patient detection.
- **F1-score:** F1-score is simply a **harmonic average of precision and recall.** F1 Score is typically used if we need to seek a balance between Precision and Recall and there is an uneven class distribution.

It must be noted that, strictly speaking, precision, recall and f1-score are defined only for binary classification. However,for

multi-class classification, we can define these metrics for each class separately. Furthermore, we can take averages of these measures across all classes to compute a global metric. This can be done through the following two metrics:

- **Macro averaging:** In this method, a simple average is done without taking into account any class imbalance.
- **Weighted averaging:** In this method, averaging is done in a weighted manner after accounting for class imbalance using support, which is simply the number of instances belonging to a particular class according to true labels.

We will be using **weighted averaged f1-score** as the metric for comparison between models. The dataset is split into training, validation and test sets. The test set is kept aside and will be used only after choosing the best model. Comparison between models is done using validation set.

A. Model 1

In this model, we allow the tree to grow without any constraints whatsoever. We observe that such a tree achieves a **training accuracy of 100%!**. This is a clear indication that the model is **overfitting.** The performance of the model on validation set is shown in Figure 10. The table reports macro averages and weighted averages. The weighted averaged f1-score is **0.956.**

	macro avg	weighted avg
precision	0.952806	0.956235
recall	0.892324	0.956647
f1-score	0.920520	0.955759

Fig. 10. Metrics for model 1

B. Model 2

In this model, we perform pruning by restricting the depth. We take the maximum allowed depth as a hyperparameter and perform a grid search on possible values. The evolution of weighted f1-score on **validation set** with tree depth is shown in Figure 11. The highest score is obtained for a **depth of 9.** The performance of the model with a depth of 9 on validation set is shown in Figure 12. The weighted averaged f1-score is **0.968. Clearly the model outperforms Model 1.**

C. Model 3

In this model, we perform bagging of decision trees. Since we are taking an ensemble, we can allow each tree to grow in an unrestricted fashion. We train each tree on a random subset of the training data, with the subset size being half of the total dataset size. We take the number of trees as a hyperparameter and perform a grid search on possible values. The evolution of weighted f1-score on **validation set** with number of trees is shown in Figure 13. The highest score is obtained for a **tree count of 50.** The performance of the model with number

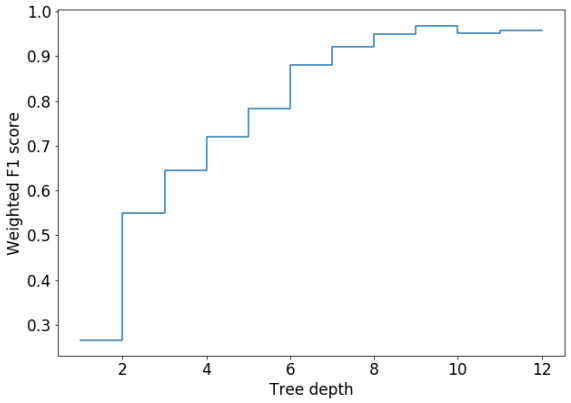

Fig. 11. Evolution of f1-score with tree depth

	macro avg	weighted avg
precision	0.900138	0.971120
recall	0.951889	0.968208
f1-score	0.921115	0.968381

Fig. 12. Metrics for model 2

of trees set to 50 on validation set is shown in Figure 14. The weighted averaged f1-score is **0.962. Clearly the model outperforms Model 1, but is not as good as Model 2.**

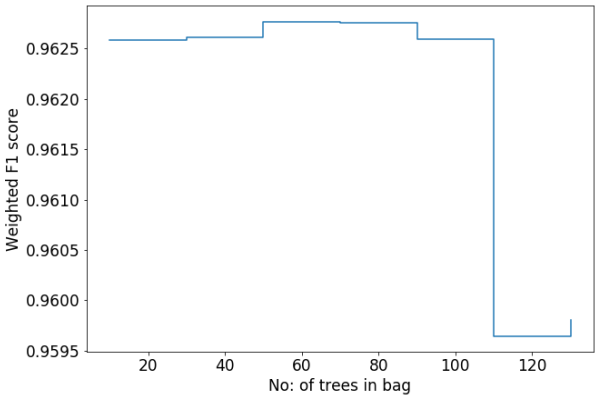

Fig. 13. Evolution of f1-score with number of trees

D. Best Model

From the above analysis, we conclude that Model 2 performs the best. Hence, we train a decision tree of depth 9 on the whole dataset consisting of training set and the validation set. We then evaluate this on the test set. The metrics for this final model is given in Figure 15 and the confusion matrix is given in Figure 16.

	macro avg	weighted avg
precision	0.921882	0.962287
recall	0.904618	0.962428
f1-score	0.911672	0.962087

Fig. 14. Metrics for model 3

To get a sense of the importance of features, we also calculate the feature importance scores of each feature using an in-built method provided by scikit-learn. Essentially, the importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance. A plot showing the feature importance scores is given in Figure 17. **It can be seen that safety rating is the most important feature, whereas number of doors is the least important feature.** It is noteworthy that this agrees with our analysis during the exploratory phase.

	macro avg	weighted avg
precision	0.896465	0.972222
recall	0.960744	0.968208
f1-score	0.922977	0.969298

Fig. 15. Metrics for the best model

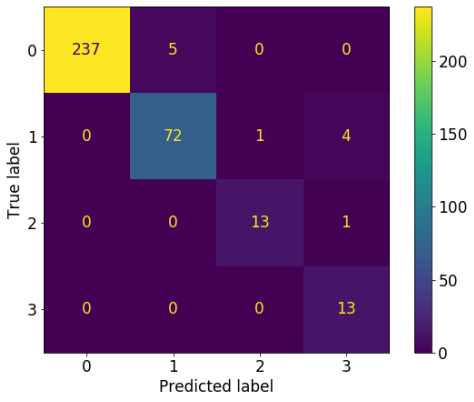

Fig. 16. Confusion matrix for the best model

Comparison with XGBoost

In almost any modern-day classification problem, the best results are often obtained by making use of boosting models. The most commonly used as well as most effective of the boosting techniques is XGBoost. By comparing the performance of our decision tree model with that of an XGBoost model, we can get a sort of benchmark on the performance.

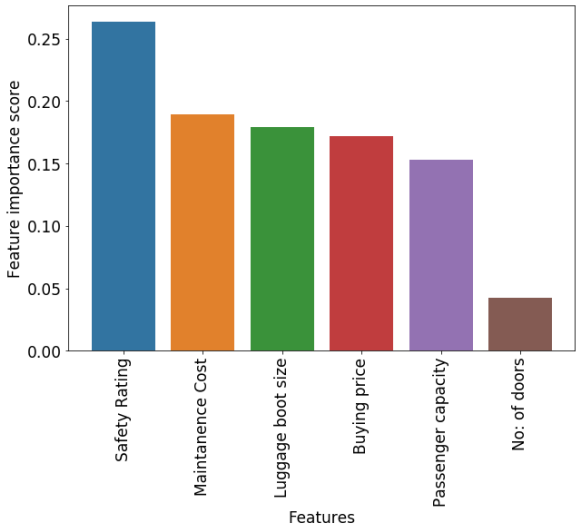

Fig. 17. Feature importance scores for the best model

XGBoost model was trained using a grid search on its hyperparameters. Cross-validation with three folds was performed for evaluating the performance. The values for the evaluation metrics and confusion matrix for the best XGBoost model are given in Figures 18 and 19 respectively. It can be seen that **the model clearly outperforms the best decision tree model.** This is expected since boosting is, by design, an ensemble of decision trees and hence superior to a single decision tree.

	macro avg	weighted avg
precision	0.978938	0.994257
recall	0.962912	0.994220
f1-score	0.970530	0.994171

Fig. 18. Metrics for XGBoost model

XGBoost also allows us to get the feature importance scores for the features. This is shown in Figure 20. Surprisingly, buying price seems to be the most important feature for this model. This feature is very closely followed by maintenance cost and safety rating. Rest of the features have an ordering similar to what we observed for the decision tree model. From the graph, it can also be seen that buying price, maintenance cost and safety rating all have very close feature importance scores. This highlights another important aspect of boosting models: **the classification decision is based on a number of features and not one feature alone. This helps enhance the robustness of the model.**

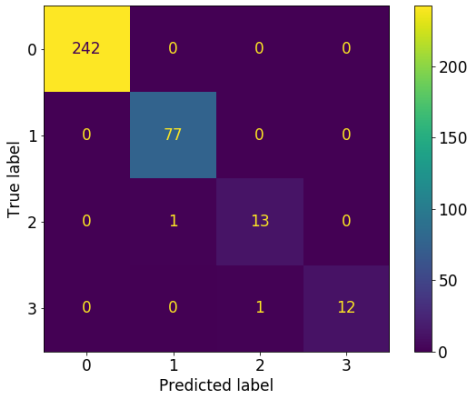

Fig. 19. Confusion matrix for XGBoost model

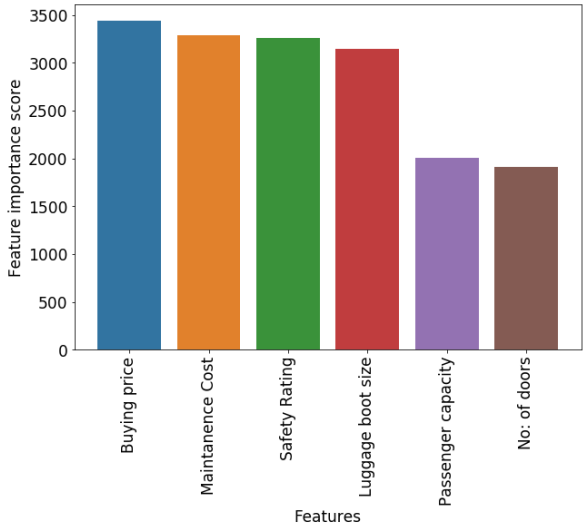

Fig. 20. Feature importance scores for XGBoost model

V. Conclusions

From our extensive analysis of the given dataset using decision tree classifier, we arrive at the following conclusions:

- Decision tree classifier, despite having a very simple structure, performs reasonably well in the classification task and mimics human decision making process.
- Decision trees are prone to overfitting. Hence, it is necessary to take remedial measures like pruning and bagging to overcome this issue.
- Accuracy may not be a good measure of performance for imbalanced classification tasks. It is necessary to use metrics like weighted f1-score for such cases.
- Best performance is obtained after using pruning and setting the maximum depth to 9.
- Safety rating is found to be the most important feature whereas number of doors is found to be the least important feature, as was also observed during the initial qualitative analysis.

VI. Avenues for further research

Decision classifier, despite working reasonably well, has a very simple structure which might not be sufficient in many cases. Using more sophisticated tree-based models like Random Forests might produce better results, though at the expense of interpretability. Doing some kind of feature preprocessing like PCA, to extract important features might also help.

References

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[5] "Decision tree classifier." https://en.wikipedia.org/wiki/Decision_tree.

A mathematical essay on random forest

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this assignment is to explore the mathematical formalism behind random forest classifier and then to use it in a real-life application. In this assignment, as a real-life application, random forest classifier is used to formally identify the factors which could have been used to predict how acceptable a car is from the popular car evaluation dataset. Data visualization, cleaning and modelling is done using Python. The analysis enables us to arrive at the conclusion that it is possible to make reasonable predictions regarding the acceptability of a car using factors including but not limited to safety rating, price and luggage boot space. This is a reworked version of the original assignment with improvements to the Modelling section in the form of enhanced benchmarking of the model including a comparison with an XGBoost model. The plots have also been improved to enhance the clarity of presentation.**

***Index Terms*—random forest, python, visualization, predictive modelling, multinomial classification**

I. Introduction

Given a set of features and a target variable, predictive modelling is typically used for generating a model which can make predictions for cases where we do not know the value of the target variable, i.e., only the features are available. Apart from this use, a model can also be used for developing an intuition of how various factors influence the target variable. In this assignment, we try to make use of a model for the purpose of identifying the key factors which influence the decision in a classification problem.

In particular, we make use of random forest classifier for the purpose of identifying relationships in a classification problem. Random forests are among the most popular machine learning algorithms as they generate reasonable predictions across a wide range of data while requiring little configuration. Random forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time. Random decision forests essentially correct for decision trees' habit of overfitting to their training set. It is a very flexible classifier in the sense that it can accommodate both continuous and categorical variables. In our particular problem, we will restrict ourselves to a classification problem using categorical variables.

In our problem setting, the goal is to use random forest classifier to predict the acceptability level of a car given a variety of factors like price, number of doors, passenger capacity, luggage boot space and safety rating. We make use of the publicly available car evaluation dataset for building the model. After building the model, we evaluate the model using a variety of evaluation metrics. By examining how well the model performs, we can identify how good the identified relationships are. Also, we check how critical each feature is to the prediction and create a ranking based on the importance of features.

Section II gives an overview of the various techniques used for data cleaning, data visualization and an initial exploratory analysis. A lot of insights can be gained just by making qualitative observations from the given data. Section III gives a short description of the mathematical formalism behind random forest. Section IV describes the various models that were tried and the results that were obtained by applying random forest in this particular case. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we describe the process of data cleaning and data visualization. We also make some qualitative observations.

A. Preliminary analysis

The given dataset has 1728 rows and 7 columns. It must be noted that the dataset itself lacked any column headings: they had to be added in manually. A brief overview of the dataset is presented in Figure 1. We observe that we have **only categorical variables.** It also seems that there are no null values.

We also look at the distribution for our target class, which is the acceptability rating termed 'target'. On plotting, we obtain Figure 2. We have four categories: unacc (Unacceptable), acc (Acceptable), good (Good) and vgood (Very good). The count of cars in the unacceptable category is disproportionately high compared to other categories. We will have to account for this during model building.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   target    1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Fig. 1. Summary of the dataset

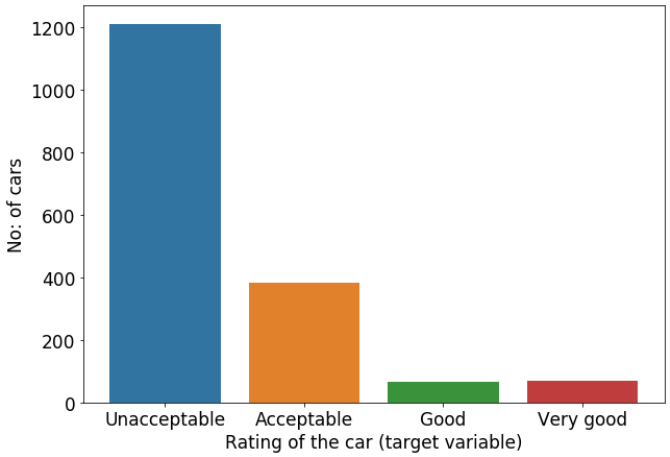

Fig. 2. Target class imbalance

B. Feature by feature analysis

Since all 6 features are categorical, for each feature we create a count plot, separating out the target classes. This can give qualitative insights regarding how a feature may affect the target variable.

Buying price

We obtain the count plot as shown in Figure 3. We make the following observations:

- Cars with high and very high buying price have a large proportion of unacceptable vehicles. Furthermore, there are no good or very good vehicles at this price range.
- Medium and low priced cars have representation from all four target categories.

Intuitively, this makes sense since people generally tend to find high prices unacceptable. For a deal to be considered good/ very good, it is almost always the case that the price should be on the lower end.

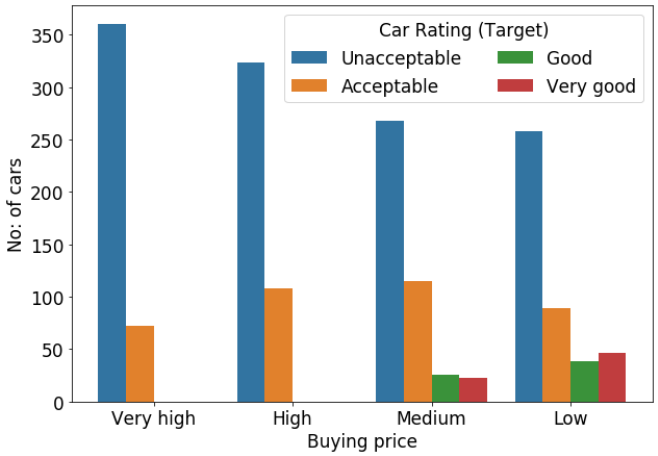

Fig. 3. Distribution of buying price

Maintenance price

We obtain the count plot as shown in Figure 4. We observe that the distribution is largely identical to the distribution for buying price. Again, we expect people to find high maintenance cost to be generally unacceptable.

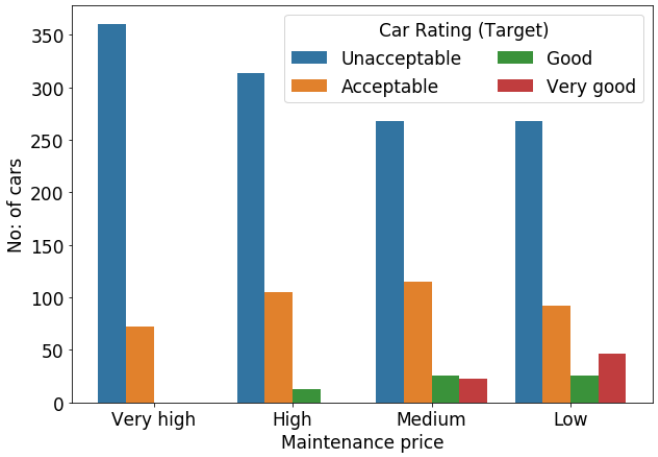

Fig. 4. Distribution of maintenance price

Number of doors

We obtain the count plot as shown in Figure 5. We observe that the distribution is largely identical for each category, i.e., irrespective of what the number of doors is, the distribution is same. In other words, the number of doors doesn't really provide any information regarding what would be the acceptability rating of a car. This implies that number of doors is not really a deciding factor as far as acceptance rating is concerned.

Passenger capacity

We obtain the count plot as shown in Figure 6. We make the following observations:

- Cars that can accommodate only two people are all unacceptable.
- Cars which can accommodate four and more than four have similar distributions.

This seems to suggest that while having at least 4 seats is essential, beyond that the number of seats doesn't really have an impact on the acceptability rating.

Size of luggage boot

We obtain the count plot as shown in Figure 7. We observe that though all categories have representation from all target categories, in general, the rating increases as the luggage boot size increases, which is what we would expect intuitively anyway.

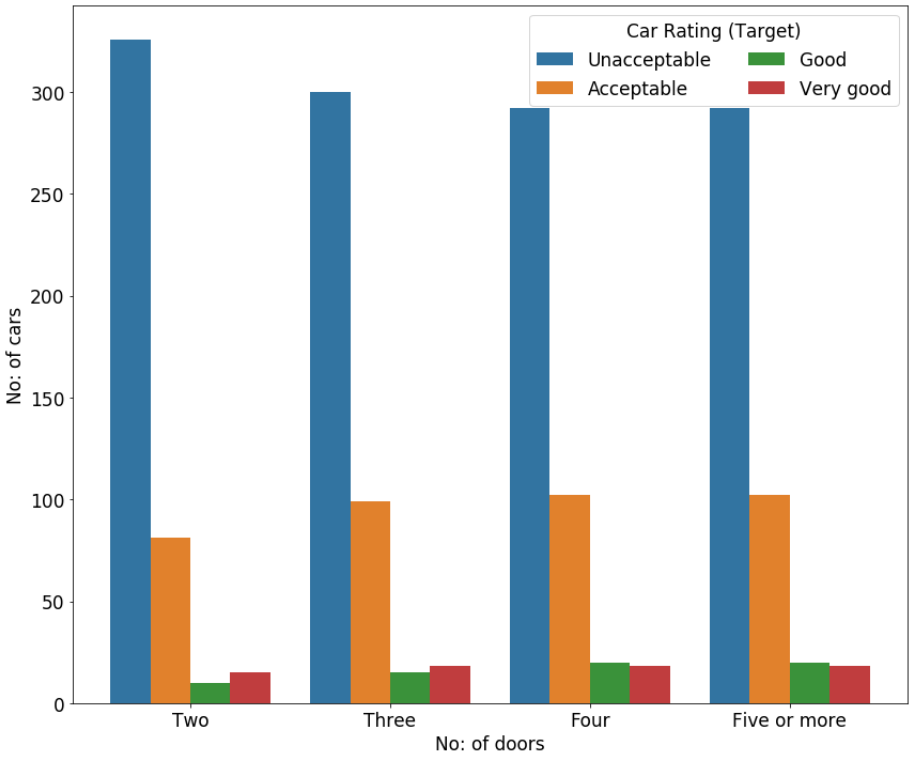

Fig. 5. Distribution of number of doors

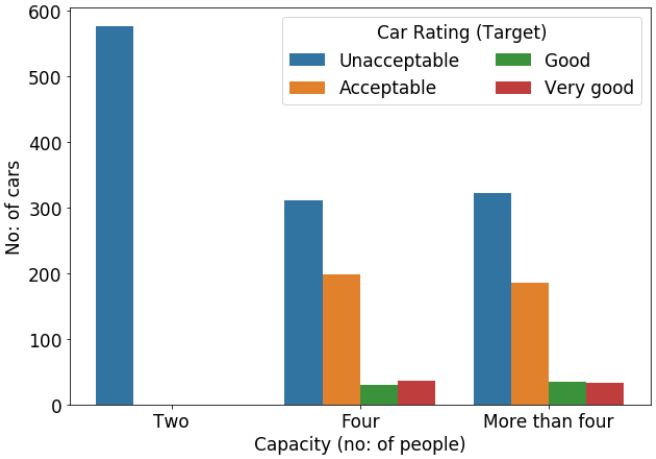

Fig. 6. Distribution of passenger capacity

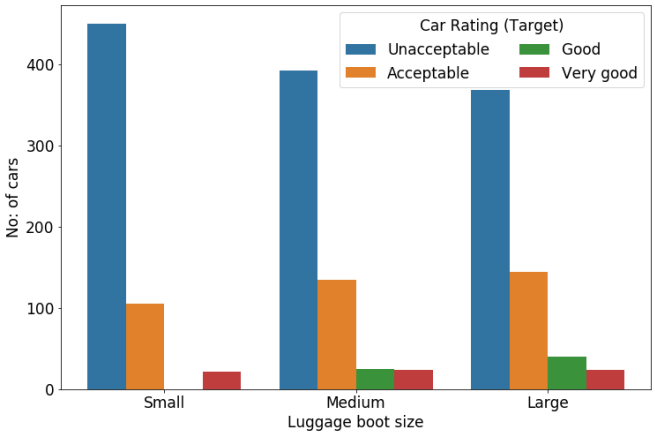

Fig. 7. Distribution of luggage boot size

Safety rating

We obtain the count plot as shown in Figure 8. We make the following observations:

- All low safety rated cars are unacceptable, which is what we would expect.
- As safety level increases, proportion of unacceptable cars decreases.

This seems to suggest that safety is a strong factor in predicting the acceptability of a car.

III. Model: Random Forest Classifier

In this section, we will give a brief overview of the mathematical formalism behind the random forest classifier.

Random forest is essentially an ensemble model made of a collection of decision trees. Decision trees have been discussed in great detail in the previous assignment. Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

The primary concern with decision trees is that they tend to overfit the data when suitable regularization techniques are

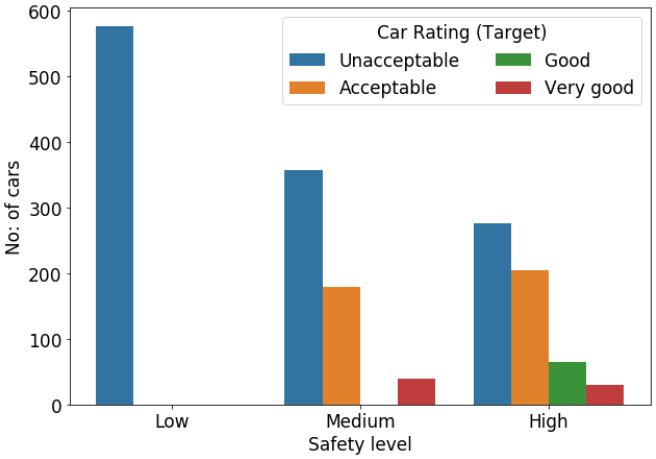

Fig. 8. Distribution of safety levels

not employed; random forest tries to overcome this problem. The idea behind random forest is simple: rather than relying on the prediction of a single tree, which can be inaccurate due to overfitting, train a large number of trees and make a decision using the predictions of all the trees. In the case of regression, this could be by making use of averaging, whereas in the case of classification this could be by taking the most voted class.

In other words, decision trees are essentially low bias, high variance classifiers. By averaging the predictions of several trees, we are bringing down the variance by incurring a marginal increase in bias. Therefore, random forests are low variance classifiers.

To make things more concrete, we consider one case where random forests easily outperform decision trees. Consider the decision boundary shown in Figure 9. Figure 10 represents the decision boundary learnt by a decision tree. Since a decision tree can only have boundary segments parallel to either x-axis or y-axis, we end up with the boundary as shown in the this figure. However, since random forests average over a number of decision trees, it comes up with the decision boundary as shown in Figure 11. Clearly, this more closely resembles the decision boundary which we would like to learn.

Fig. 9. Decision boundary to be learned

The major benefits offered by a random forest classifier are:

- **Robust model:** The greatest advantage of a random forest model is that it is very robust. This means that it achieves very good performance on a variety of datasets with very minimal tuning. In many cases, random forest models can be used out-of-the-box, with very minimal hyperparameter tuning.
- **Non-parametric approach**: Makes no assumptions of the training data or prediction residuals; e.g., no distributional, independence, or constant variance assumptions.
- **Requires little data preparation**: Other techniques often require data normalization. Since trees can handle qualitative predictors, there is no need to create dummy variables

Fig. 10. (Decision boundary learned by decision tree

Fig. 11. Decision boundary learned by random forest

Some limitations of this approach:

- **Black-box model**: Although moving to a random forest from decision tree greatly improves model performance, this is achieved by sacrificing interpretability. Unlike decision trees, which are considered white-box models, random forests are black-box models in the sense that their results may not be easily explainable.
- **Non-optimality**: Random forest learning algorithms are based on heuristics such as the greedy algorithm where locally optimal decisions are made at each node. This is because since random forests involve a large number of individual decision trees, it is not possible to train each tree optimally as this is computationally infeasible. The downside of heuristic algorithms is that they cannot guarantee globally solutions.

A. Mathematical formalism

A random forest is essentially an ensemble of decision trees. The output of a random forest can hence be thought of as some aggregation of the output of individual trees. Therefore, the mathematical principles underlying random forests are same as that of decision trees. A brief review of the mathematical formalism behind decision tree is given below.

What a decision tree essentially learns is a decision for each decision node. In particular, suppose a decision tree makes use of an impurity measure S (various measures for quality are

discussed later). Suppose a node splits the dataset into two subsets left and right. Then the node computes the quantity:

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

where m_{left} is the number of samples in the left subset, m_{right} is the number of samples in the right subset, S_{left} is the impurity measure for the left subset and S_{right} is the impurity measure for the right subset. The attempt is to find the split which minimizes $J(k, t_k)$.

We can regard $J(k, t_k)$ as the 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 cut-point in order to split the data further so as to minimise the cost function. This process continues till the set stopping criterion is reached or when every training sample is perfectly classified.

The above algorithm is repeated for a number of trees. The ensemble of trees are then used for making a prediction. Based on whether we sample the features themselves for each tree or not, we have two approaches: bagging and random forests. These will be examined in more detail in a later section.

An overview of two common attribute selection measures/ impurity measures is given below:

1) Gini Index/Impurity: This is essentially a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p_{i,k}}(1 - \hat{p_{i,k}})$$

where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node. The Gini index takes on a small value if all of the $\hat{p}_{i,k}$'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. The equation for Gini index can also be rewritten as:

$$G = 1 - \sum_{k=1}^{K} \hat{p_{i,k}^2}$$

2) Entropy: The formulation has a close resemblance with the entropy formulation in thermodynamics.

$$D = -\sum_{k=1}^{K} \hat{p_{i,k}} \log \hat{p_{i,k}}$$

where again K is the total number of classes and $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node. As was the case with Gini index, a smaller value of entropy signifies higher purity and a larger value of entropy signifies impurity.

B. Ensemble methods

As discussed above, there are two approaches to creating an ensemble of decision trees: bagging and random forests. These are described below:

1) Bagging: Decision trees suffer from high variance and can be very non-robust, in general. This means that even a slight change or using a different set of samples 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, i.e. we take repeated samples from the same training data set. Hence, we can sort of obtain a proxy for distinct datasets by repeatedly re-sampling observations from the original dataset. This sampling is done with replacement, meaning the same observation can occur in more than one of such synthetically generated datasets.

To apply bagging to trees, we simply construct B trees $f^1(x), f^2(x), .., f^B(x)$ using B bootstrapped training sets and average the resulting predictions to obtain a single low variance statistical model. The constructed trees can be allowed to grow deep even if this means that they suffer from high variance, since averaging reduces variance and hence gives a better model. For classification, instead of averaging we take the most frequent prediction. Rest of the approach remains the same. B here would be a hyperparameter. High values of B just implies we are averaging over many trees and hence apart from computational complexity, increasing B doesn't lead to overfitting. In practice, B is kept sufficiently large so that the error settles down.

2) Random Forest: 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 in fact a hyper-parameter that could be chosen via cross validation.

The main motivation behind random forest is the fact that prediction from individual trees of a bagged model tend to be correlated. Random forests help decorrelate the trees by making use of random sampling of features. The variance after averaging reduces in a much better fashion if the predictions from each tree are uncorrelated. The main advantage of bagging/random forest is that all the trees can be trained in parallel and hence could be sped up easily by making use of some form of parallel training. But this kind of approach cannot be readily employed in sequentially/adaptively trained models, i.e. boosting based approaches.

C. Time complexity

In a decision tree, a split has to be found until a maximum depth d has been reached. In the worst case, maximum depth is equal to n, the training set size. The strategy for finding a split is to look for a suitable threshold for each feature of the dataset. Since there are p such variables, we make np decisions per node. Considering a random forest with n_{trees} such trees:

- Assuming no-feature selection, the upper bound for train time complexity is $O(n_{\text{trees}} n^2 p)$.

- If $\sqrt{p}$ random feature selection is done at each node, the upper bound for train time complexity is $O(n_{\text{trees}} n^2 \sqrt{p})$.

IV. Modelling

In this section, we discuss the application of the random forest classifier to our problem.

We create and evaluate two models:

1) In Model 1, we do not do random sampling of features, i.e., we perform bagging.
2) In Model 2, we perform random sampling of features.

For each model, we take care to account for the inherent imbalance in the training dataset with respect to the target class. This is done by making use of appropriate weights during the training of each decision tree of the ensemble model.

We evaluate the models based on multiple metrics. A short description of the metrics used is given below:

- **Accuracy:** Accuracy is simply the **ratio of number of correct predictions to total number of predictions.** Although this seems like a very good metric intuitively, accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on datasets with an equal class distribution.
- **Precision:** Precision is the **ratio of true positives to the total positive predictions.** Precision is typically used when the cost of false positive is high. For instance, email spam detection.
- **Recall:** Precision is the **ratio of true positives to the total positive ground truths.** Recall is typically used when the cost of false negative is high. For instance, in fraud detection or sick patient detection.
- **F1-score:** F1-score is simply a **harmonic average of precision and recall.** F1 Score is typically used if we need to seek a balance between Precision and Recall and there is an uneven class distribution.

It must be noted that, strictly speaking, precision, recall and f1-score are defined only for binary classification. However,for multi-class classification, we can define these metrics for each class separately. Furthermore, we can take averages of these measures across all classes to compute a global metric. This can be done through the following two metrics:

- **Macro averaging:** In this method, a simple average is done without taking into account any class imbalance.
- **Weighted averaging:** In this method, averaging is done in a weighted manner after accounting for class imbalance using support, which is simply the number of instances belonging to a particular class according to true labels.

We will be using **weighted averaged f1-score** as the metric for comparison between models. The dataset is split into training, validation and test sets. The test set is kept aside and will be used only after choosing the best model. Comparison between models is done using validation set.

A. Model 1

In this model, we create a bagging model. The number of trees to be included is a hyperparameter. Essentially, we keep increasing the number of trees till the metric (weighted f1-score in our case) stabilizes. The plot of score against number of trees is shown in Figure 12. Clearly, **the score stabilizes for a tree count of more than 400.** Hence, we set the number of trees equal to 400. Also note that we perform bootstrapping to create training sets for each individual tree. Two-thirds of the total number of datapoints is used for training each tree. The performance of the model on validation set is shown in Figure 13. The table reports macro averages and weighted averages. The weighted averaged f1-score is **0.959.**

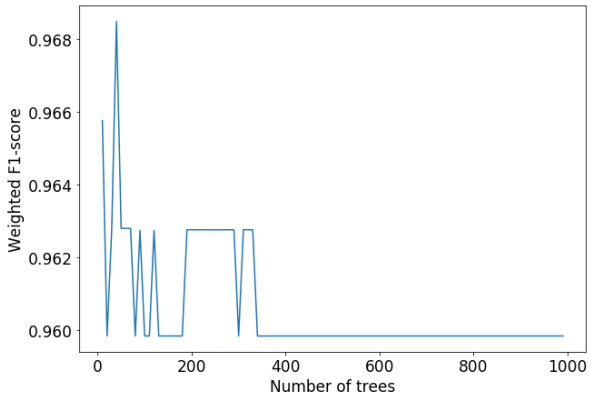

Fig. 12. Evolution of f1-score with number of trees

	macro avg	weighted avg
precision	0.922041	0.959164
recall	0.915982	0.959538
f1-score	0.918912	0.959244

Fig. 13. Metrics for model 1

B. Model 2

In this model, we create a random forest model, with feature sampling. The number of trees is set to be 400 (from the above analysis). In this case, the maximum number of features to be included while random sampling is taken as the hyperparameter. In our case, the total number of available features is only 6. Hence, we can perform a simple grid search from 1 through 6 for the number of features to be used while random sampling. The plot of score against number of features is shown in Figure 14. Clearly, **the score achieves a peak for the number of features equal to 2.** Hence, we set the number of features during sampling equal to 2. Also note that we perform bootstrapping to create training sets for each individual tree. Two-thirds of the total number of datapoints is used for training each tree. The performance of the model on

validation set is shown in Figure 15. The table reports macro averages and weighted averages. The weighted averaged f1-score is **0.968. Clearly the model outperforms Model 1.**

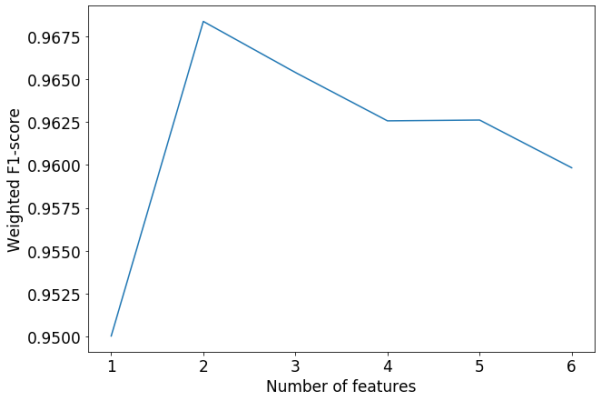

Fig. 14. Evolution of f1-score with number of features

	macro avg	weighted avg
precision	0.941189	0.968033
recall	0.925722	0.968208
f1-score	0.933149	0.968068

Fig. 15. Metrics for model 2

C. Best Model

From the above analysis, we conclude that Model 2 performs the best. Hence, we train a random forest model with a random sampling of 2 features and 400 trees on the whole dataset consisting of training set and the validation set. We then evaluate this on the test set. The metrics for this final model is given in Figure 16 and the confusion matrix is given in Figure 17.

To get a sense of the importance of features, we also calculate the feature importance scores of each feature using an in-built method provided by scikit-learn. Essentially, the importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance. A plot showing the feature importance scores is given in Figure 18. **It can be seen that safety rating is the most important feature, whereas number of doors is the least important feature.** It is noteworthy that this agrees with our analysis during the exploratory phase.

Comparison with XGBoost

In almost any modern-day classification problem, the best results are often obtained by making use of boosting models. The most commonly used as well as most effective of the boosting techniques is XGBoost. By comparing the performance of our random forest model with that of an XGBoost model, we can get a sort of benchmark on the performance.

	macro avg	weighted avg
precision	0.959375	0.986597
recall	0.958439	0.985549
f1-score	0.956164	0.985557

Fig. 16. Metrics for the best model

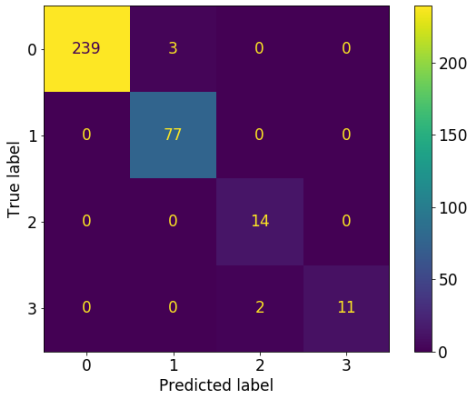

Fig. 17. Confusion matrix for the best model

XGBoost model was trained using a grid search on its hyperparameters. Cross-validation with three folds was performed for evaluating the performance. The values for the evaluation metrics and confusion matrix for the best XGBoost model are given in Figures 19 and 20 respectively. It can be seen that **the model clearly outperforms the best random forest model.**

XGBoost also allows us to get the feature importance scores

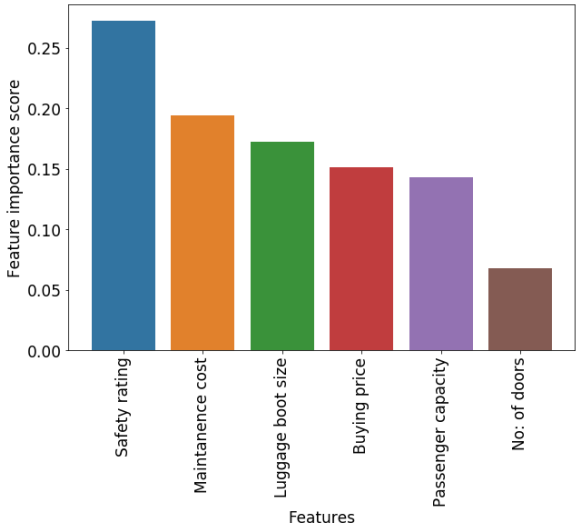

Fig. 18. Feature importance scores

	macro avg	weighted avg
precision	0.978938	0.994257
recall	0.962912	0.994220
f1-score	0.970530	0.994171

Fig. 19. Metrics for XGBoost model

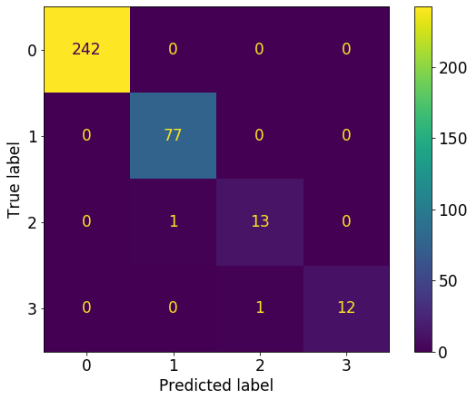

Fig. 20. Confusion matrix for XGBoost model

for the features. This is shown in Figure 21. Surprisingly, buying price seems to be the most important feature for this model. This feature is very closely followed by maintenance cost and safety rating. Rest of the features have an ordering similar to what we observed for the decision tree model. From the graph, it can also be seen that buying price, maintenance cost and safety rating all have very close feature importance scores. This highlights another important aspect of boosting models: **the classification decision is based on a number of features and not one feature alone. This helps enhance the robustness of the model.**

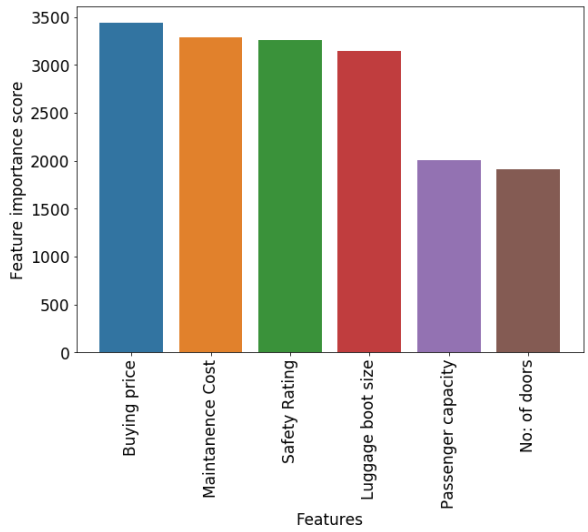

Fig. 21. Feature importance scores for XGBoost model

V. Conclusions

From our extensive analysis of the given dataset using random forest classifier, we arrive at the following conclusions:

- Random forest classifier performs very well on the given dataset. In fact, it performs better than the decision tree classifier which we created in the previous assignment.
- Although random forest model performs better than the decision tree model, this improvement comes at the cost of reduced interpretability.
- Accuracy may not be a good measure of performance for imbalanced classification tasks. It is necessary to use metrics like weighted f1-score for such cases.
- Best performance is obtained by setting the number of randomly selected features to 2 and number of trees to 400.
- Safety rating is found to be the most important feature whereas number of doors is found to be the least important feature, as was also observed during the initial qualitative analysis.

VI. Avenues for further research

Random forest classifier performs quite well for the given dataset. Doing some kind of feature preprocessing like PCA, to extract important features before inputting it to the model might help. Trying out boosting methods might also help improve the performance of the model.

References

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[5] "Random forest classifier." https://en.wikipedia.org/wiki/Random_forest.

A mathematical essay on support vector machine

Gautham Govind A
Dept. of Electrical Engineering
Indian Institute of Technology Madras
ee19b022@smail.iitm.ac.in

***Abstract*—The objective of this assignment is to explore the mathematical formalism behind support vector machine and then to use it in a real-life application. In this assignment, as a real-life application, support vector machine is used to formally identify how certain features could be used to predict whether a star is a pulsar star or not from the given pulsar star dataset. Data visualization, cleaning and modelling is done using Python. The analysis enables us to arrive at the conclusion that it is possible to make reasonable predictions regarding whether a star is a pulsar or not using simple statistics of the integrated pulse profile and the DM-SNR curve.**

***Index Terms*—support vector machine, python, visualization, predictive modelling, binary classification**

I. Introduction

Given a set of features and a target variable, predictive modelling is typically used for generating a model which can make predictions for cases where we do not know the value of the target variable, i.e., only the features are available. Apart from this use, a model can also be used for developing an intuition of how various factors influence the target variable. In this assignment, we try to make use of a model for the purpose of identifying the key relationships which influence the decision in a classification problem and then use this model for making predictions.

In particular, we make use of support vector machine (SVM) for the purpose of identifying relationships in a classification problem. Support vector machine is a very robust machine learning algorithm which can be used for tackling both classification and regression problems. Support vector machine is quite popular in the machine learning community because of its strong theoretical framework based on statistical learning theory . Although it is a linear classifier by default, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. This is what makes them really powerful. In our particular problem, we will be using support vector machines for classification.

In our problem setting, the goal is to use support vector machine to predict whether the given star is a pulsar or not given a variety of simple statistics of the integrated pulse profile and the DM-SNR curve. . We make use of a publicly available pulsar star dataset for building the model. After building the model, we evaluate the model using a variety of evaluation metrics. By examining how well the model performs, we can identify how good the identified relationships are. Finally, we make use of our best model to make predictions for cases where the output label is not known,

Section II gives an overview of the various techniques used for data cleaning, data visualization and an initial exploratory analysis. A lot of insights can be gained just by making qualitative observations from the given data. Section III gives a short description of the mathematical formalism behind support vector machine. Section IV describes the various models that were tried and the results that were obtained by applying support vector machine in this particular case. Section V gives a summary of the major conclusions drawn from the analysis.

II. Exploratory Data Analysis

In this section, we describe the process of data cleaning and data visualization. We also make some qualitative observations.

A. Preliminary analysis

The given dataset has 12528 rows and 9 columns. Eight of the columns form the features, whereas the ninth column is the target variable. We observe that we have **only continuous variables,** once we exclude the target variable.

We observe that the eight features are essentially mean, standard deviation, excess kurtosis and skewness of the integrated pulse profile and the DM-SNR curve. A few key points regarding excess kurtosis and skewness:

- Kurtosis represents how fat a distribution's tail is when compared to the center of the distribution.
- Excess kurtosis compares the kurtosis coefficient of the distribution with that of a normal distribution.
- Normal distributions have a kurtosis of three. Excess kurtosis can, therefore, be calculated by subtracting kurtosis by three.
- When excess kurtosis is positive, the tails are heavier and when it is negative, tails are thinner than the normal distribution.
- Skewness is a measure of degree of asymmetry observed in a probability distribution.
- Positive/Right skewed would imply that the mean of the data is greater than the median.
- Negative/Left skewed would imply that the mean of the data is less than the median.

It also seems that there are null values in three of the features. More precisely, we see that the features 'Excess kurtosis of the integrated profile', 'Standard deviation of the DM-SNR curve' and 'Skewness of the DM-SNR curve'.

We first plot the non-null values for excess kurtosis of the integrated profile. We obtain a plot as shown in Figure 1. We observe that the tails are rather thin. Hence, we use mean imputing. After imputation, we obtain the plot as shown in Figure 2. On plotting the non-null values for standard deviation of DM-SNR curve, we obtain the plot as shown in Figure 3. As the right tail is rather thick, we use median imputation. We obtain the final plot as Figure 4. Finally, we look at the plot for skewness of the DM-SNR curve. The plot is shown in Figure 5. Again, since the tail is thick we apply median imputation, giving us the plot in Figure 6.

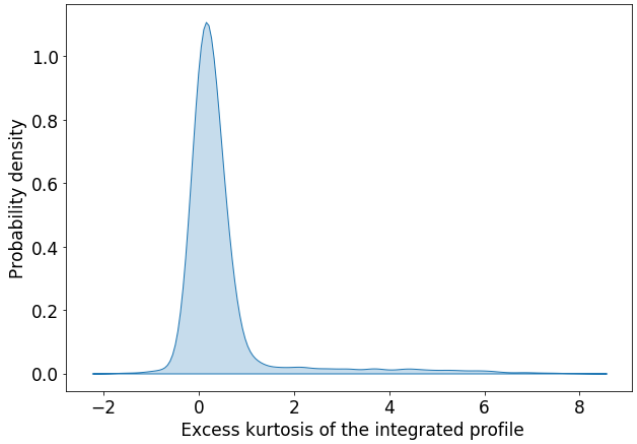

Fig. 1. Density plot (before imputation - Only non-null values)

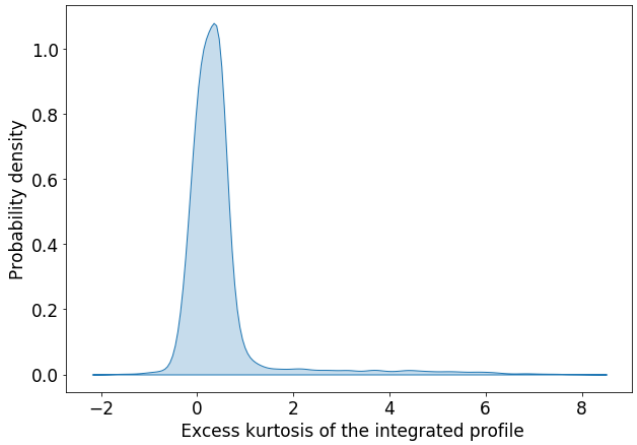

Fig. 2. Density plot (after imputation)

We also look at the distribution for our target class, which is whether a star is a pulsar or not. On plotting, we obtain Figure 7. It is observed that only 9.20% of the total datapoints belong to the class of pulsar stars, implying the dataset is skewed. We will have to account for this during model building.

B. Feature by feature analysis

For each feature, we generate a Kernel Density Estimate (KDE) plot, which essentially gives the probability density plot for that particular feature. Moreover, we make the plots

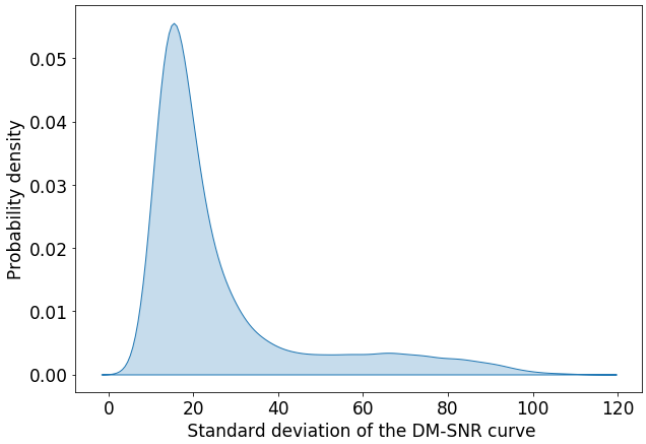

Fig. 3. Density plot (before imputation - only non-null values)

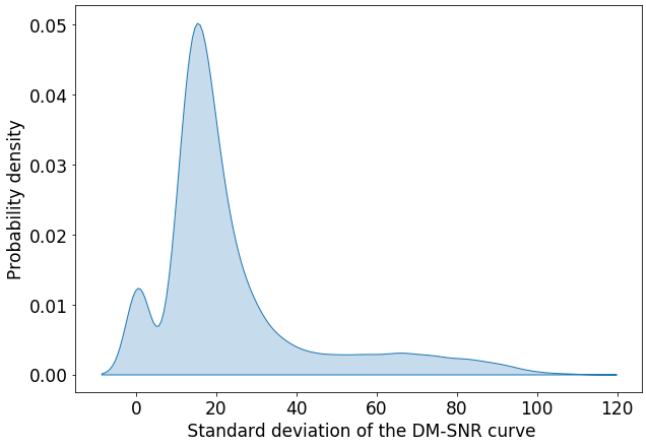

Fig. 4. Density plot (after imputation)

separately for stars which are categorized as pulsars and those which are not so as to gain some qualitative idea of how the feature is distributed for the two categories.

The plots for the mean and standard deviation of the integrated profile are shown in Figures 8 and 9 respectively. **It can be seen that for stars which are not pulsars, the mean and standard deviation of the integrated profile tend to be higher.** Further, the means of integrated profiles of stars which are pulsars are spread more as compared to non-pulsars.

The plots for the excess kurtosis and skewness of the integrated profile are shown in Figures 10 and 11 respectively. **In both cases, the distribution is more spread out for stars which are pulsars.**

The plots for the mean and standard deviation of the DM-SNR curve are shown in Figures 12 and 13 respectively. **It can be seen that for stars which are pulsars, the mean and standard deviation of the DM-SNR curve tend to be higher.** Further, the means and standard deviations of DM-SNR curves of stars which are pulsars are spread more as compared to non-pulsars.

The plots for the excess kurtosis and skewness of the DM-

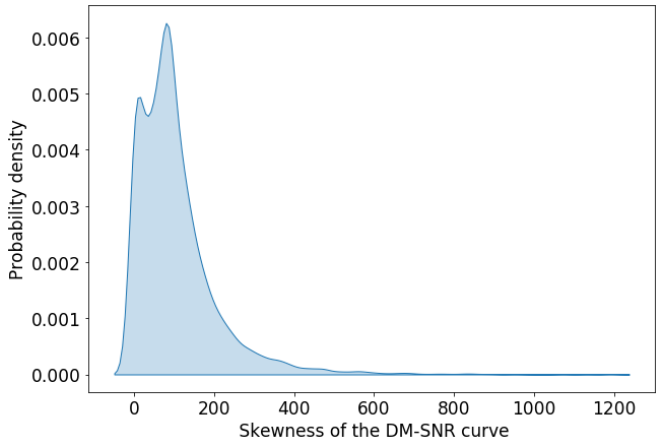

Fig. 5. Density plot (before imputation - only non-null values)

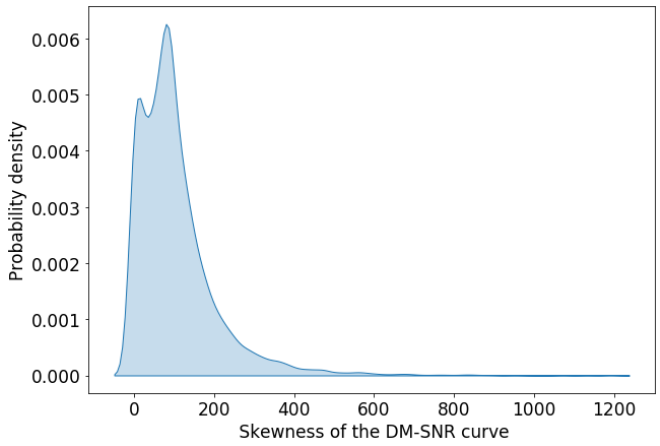

Fig. 6. Density plot (after imputation)

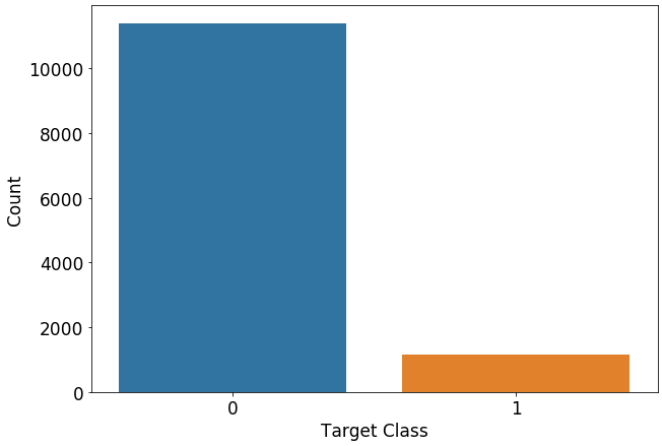

Fig. 7. Target class imbalance

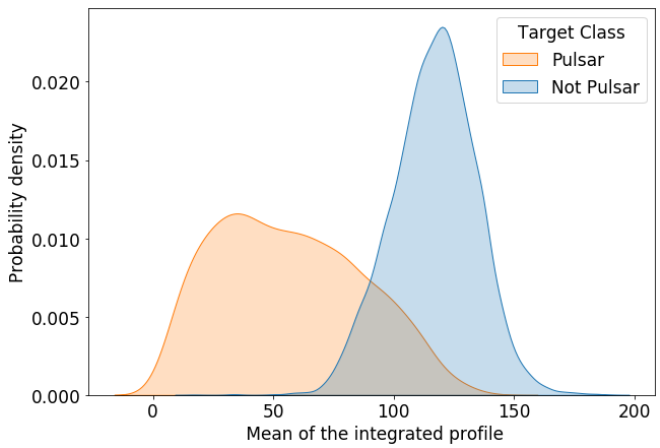

Fig. 8. Density plot for mean of integrated profile

SNR curve are shown in Figures 14 and 15 respectively. **In both cases, the distribution is more spread out for stars which are not pulsars.**

III. Model: Support Vector Machine

In this section, we will give a brief overview of the mathematical formalism behind the support vector classifier.

Support vector machines in general support both regression and classification. The classifier based on the principle of support vector machines is often referred to as support vector classifiers. By default, SVM is designed for the case of binary classification. Extension to multinomial classification is non-trivial and is not of interest to us currently. Also, SVM without any modification is a linear classifier. However, it is possible to introduce non-linearities through the 'kernel' trick which we will get to very soon.

SVMs follow a discriminant modelling approach, which means that the model directly maps each sample to a class instead of assigning class probabilities. This is in contrast with generative models like logistic regression and naive bayes which rely on computation of posterior probabilities for assigning class labels.

The support vector machine is a generalization of a simple and intuitive classifier called the maximal margin classifier. The main issue with the vanilla maximal margin classifier is that it requires the dataset to be strictly linearly separable which is a rather unreasonable requirement. SVM is an extension of maximal margin classifier which relaxes this requirement. We explore these concepts more rigorously in the following sections.

A. Maximal margin classifier

Before introducing the maximal margin classifier, it is necessary to understand what a hyperplane is. A hyperplane is a subspace of one dimension less than of its ambient space. Intuitively, it can be thought of as an extension of a line in X-Y plane or a plane in X-Y-Z plane to arbitrarily higher number of dimensions. Mathematically, a hyperplane in a p-dimensional space is governed by an equation of the form:

$$f(\mathbf{X}) = w_1x_1 + w_2x_2... + w_px_p + b = 0$$

Suppose we have a dataset with n datapoints with each datapoint having p features each. If we assume the labels to

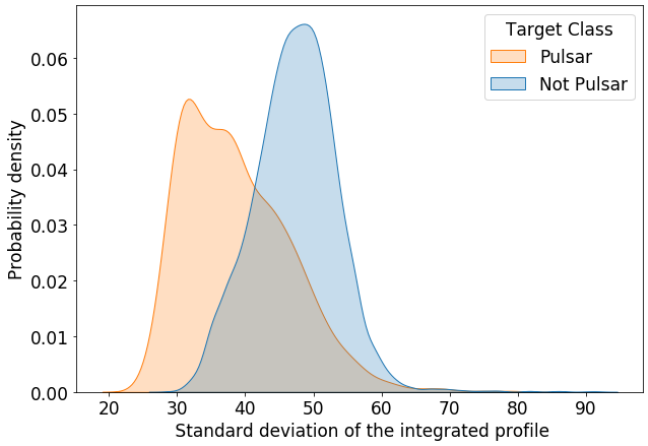

Fig. 9. Density plot for standard deviation of integrated profile

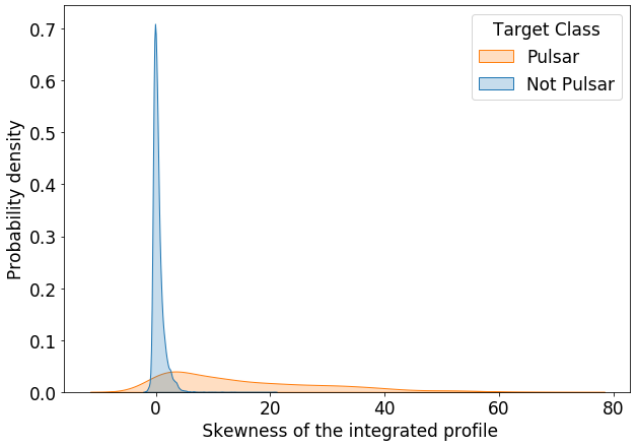

Fig. 11. Density plot for skewness of integrated profile

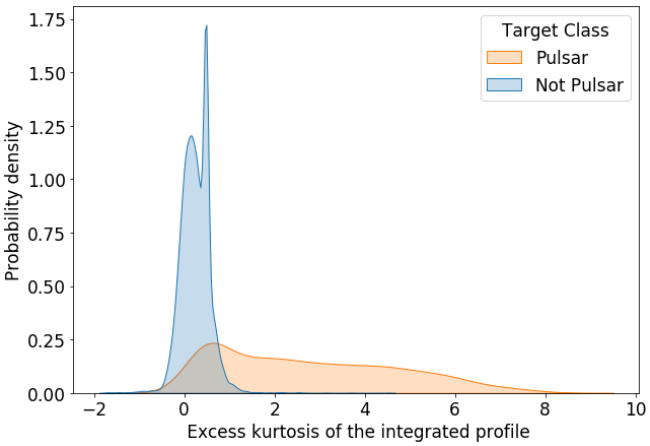

Fig. 10. Density plot for excess kurtosis of integrated profile

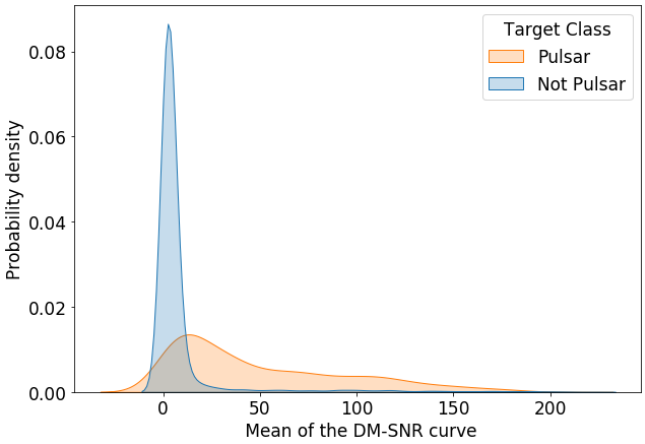

Fig. 12. Density plot for mean of DM-SNR curve

be $y_i = \{-1, +1\}$, then for the dataset to be linearly separable,

$$f(\mathbf{X}) > 0 \text{ if } y_i = +1$$

$$f(\mathbf{X}) < 0 \text{ if } y_i = -1$$

This can be rewritten as:

$$y_i f(\mathbf{X}) > 0$$

$$y_i(w_1x_1 + w_2x_2 ... + w_px_p + b) > 0$$

The condition which we have derived so far is the condition necessary for the dataset to be considered linearly separable. Now, if we know a given dataset is linearly separable, it is possible to draw several lines separating the two classes. How do we decide which line is the best?

An intuitive answer is to take the line with the largest margin, which is essentially the distance to the closest point. This strategy is what is called maximal margin classification. A key feature of maximal margin classifiers are support vectors. Essentially, the position and slope of the line is determined only by those points which are closest to the line. If these points were to shift around slightly, the line would get shifted, whereas changing points other than the supports would have no effect on the line whatsoever.

Mathematically, this can be stated as:

$$\max_{w,b} \min_{i \in n} \frac{|w^T x_i + b|}{||w||}$$

$$\text{s.t. } \text{sign}(w^T x_i + b) = y_i$$

After some mathematical manipulations, it can be shown that the above maximization is equivalent to the following minimization:

$$\min_{w.b} \frac{1}{2}||w||^2$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1$$

B. Soft margin classifier

As stated earlier, the maximal margin classifier is not ideal since it enforces linear separability, which is not compatible with most of the real-life datasets. In order to obtain a greater robustness we aim to modify the classifier to work in situations where a majority of the points are separable whereas a small

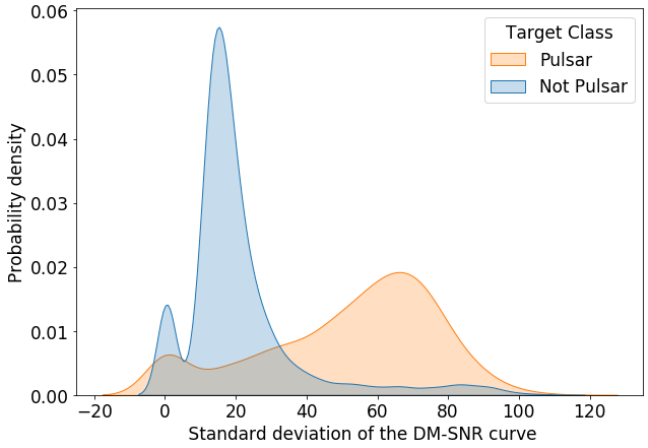

Fig. 13. Density plot for standard deviation of DM-SNR curve

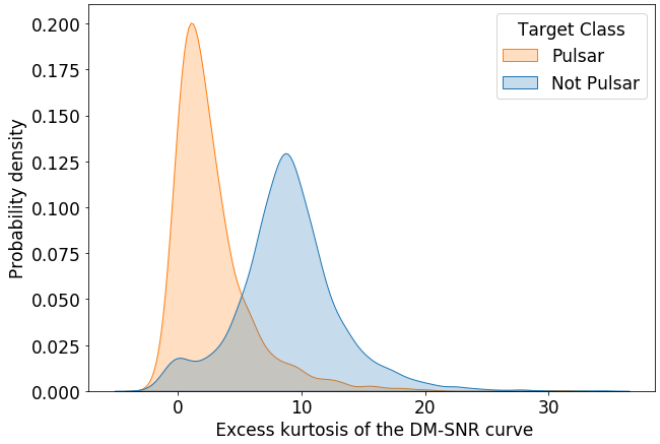

Fig. 14. Density plot for excess kurtosis of DM-SNR curve

fraction is not. This is the principle behind support vector classifier also known as the soft margin classifier.

The classifier is called soft margin because the separability condition 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 outside the margin, we instead allow some observations to be inside the margin or even on the incorrect side of the hyperplane. The points that fall on the incorrect side of the hyperplane are miss-classified. These points together with those that fall within and on the margin are the support vectors of the classifier that influence the decision boundary.

To make things more concrete mathematically, let us define the slack variable ϵ_i. If a datapoint X_i falls outside the margin, $\epsilon_i = 0$, whereas $\epsilon_i > 0$ for support vectors. The soft-margin problem could then be written as:

$$\min_{w.b} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\epsilon_i$$

$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \epsilon_i$$

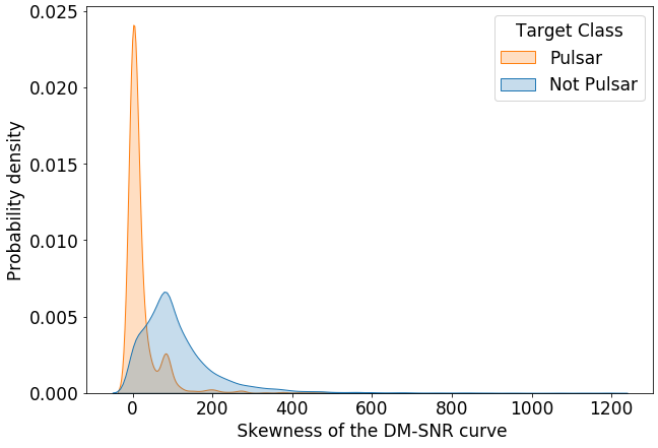

Fig. 15. Density plot for skewness of DM-SNR curve

$$\epsilon_i \geq 0$$

The hyperparameter C determines the severity of the violations to the margin. If $C \to \infty$, the cost associated for miss-classification is very high and thence results resemble that of a hard-margin classifier. If C is very less, we allow more observations to violate the margin, hence also allowing more miss-classifications.

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 but high variance. On the other hand, when C is small, the margin is wider, and we allow more violations to it. This amounts to fitting the data less forcefully and obtaining a classifier that is potentially more biased but has a lower variance. Hence we can say that C controls bias-variance trade-off.

C. Solution to the optimization problem

The Lagrangian of the soft margin classifier optimization problem in the primal space can be written as:

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

where α_i and β_i are lagrange multipliers and hence are non-negative. The above problem can be solved in the dual space by applying min-max theorem. Though the solution process is a bit complicated, 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.

D. Kernel Trick

Since 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 $K(x, x_i)$ represents the dot product in some higher dimensional plane. The intuition behind this is that we now would be able to achieve the effect of transforming X to a higher dimensional subspace without actually performing the transformation. This is because we effectively need only the dot product which can be computed using functions called kernels, which is denoted by K here.A kernel essentially is a transformation to a higher dimensional subspace through dot products without performing the transformation directly. 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 (Transformation into polynomial space of degree d).
- $K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2)$: RBF Kernel. (Transformation into infinite-dimensional Hilbert Space).

IV. Modelling

In this section, we discuss the application of the support vector classifier to our problem.

We create and evaluate multiple models based on the following hyperparameters:

- **Kernel:** We try SVC classifiers with linear and rbf kernels.
- **C:** C here is the hyperparameter which was discussed during discussion of soft-margin classifier.
- γ: γ is a hyperparameter applicable only for the case of rbf kernel. This hyperparameter helps tune the amount of regularization.

We also define each SVC classifier such that appropriate class weights are assigned to each of the target class, such that the imbalance in the dataset is accounted for.

We evaluate the models based on multiple metrics. A short description of the metrics used is given below:

- **Accuracy:** Accuracy is simply the **ratio of number of correct predictions to total number of predictions.** Although this seems like a very good metric intuitively, accuracy fails on classification problems with a skewed class distribution because of the intuitions developed by practitioners on datasets with an equal class distribution.
- **Precision:** Precision is the **ratio of true positives to the total positive predictions.** Precision is typically used when the cost of false positive is high. For instance, email spam detection.
- **Recall:** Precision is the **ratio of true positives to the total positive ground truths.** Recall is typically used when the cost of false negative is high. For instance, in fraud detection or sick patient detection.
- **F1-score:** F1-score is simply a **harmonic average of precision and recall.** F1 Score is typically used if we need to seek a balance between Precision and Recall and there is an uneven class distribution.

TABLE I
METRICS FOR THE BEST MODEL

Metric	Score
Accuracy	0.966
Precision	0.781
Recall	0.884
F1 Score	0.829

We will be using **f1-score** as the metric for comparison between models. The dataset is split into training and validation sets. The test set, which has been provided separately, is kept aside and will be used only after choosing the best model. Comparison between models is done using validation set.

After performing a grid search over the hyperparameter space and comparing multiple different models through cross-validation, we arrive at the conclusion that for the given dataset, the best set of hyperparameters are **kernel = rbf, C = 5.039 and** $\gamma = 0.231$. The best model has 1090 support vectors for class 0 and 224 support vectors for class 1. **This means that out of the total 8769 samples, only 1314 are support vectors.** This goes to show that only a small fraction of the points actually contribute to the SVM decision boundary.

The confusion matrix for the best model is shown in Figure 16. The values for the evaluation metrics are given in Table I.

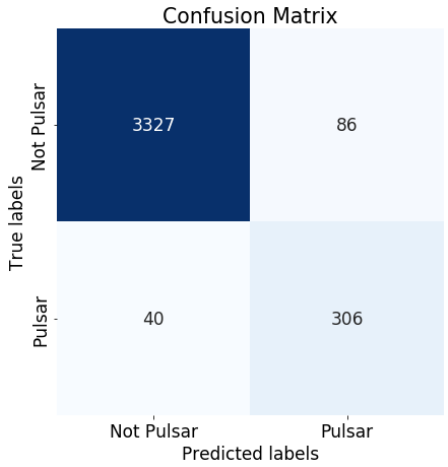

Fig. 16. Confusion matrix for the best model

ROC curve is another common tool used with binary classifiers. A Receiver Operating Characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various

threshold settings. The ROC curve for our model is given in Figure 17.

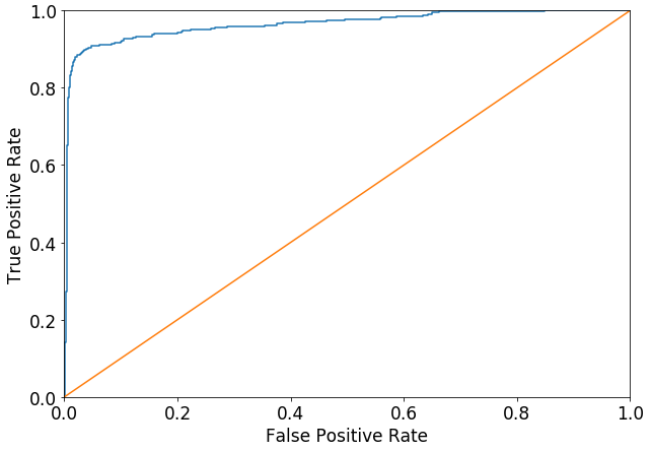

Fig. 17. ROC curve for the best model

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 classifier is 0.964.**

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 threshold. But since SVM is defined in a such a way that it doesn't produce probabilities directly, we can only approximate these by computing signed distance between the hyperplane and the points. So the above is only an estimate of the ROC curve.

Finally, predictions were also made for the best model that was obtained on the given test dataset.

V. Conclusions

From our extensive analysis of the given dataset using support vector classifier, we arrive at the following conclusions:

- Support vector machines are robust algorithms which are. in comparison. less sensitive to outliers than most of the other popular algorithms.
- Usage of the kernel trick is a clever way to search in higher dimensional space without increasing computational complexity.
- Accuracy may not be a good measure of performance for imbalanced classification tasks. It is necessary to use metrics like weighted f1-score for such cases.
- Best performance is obtained by setting kernel as rbf, C = 5.039 and $\gamma = 0.231$.
- A good fit is obtained for the model, meaning whether a star is a pulsar or not can be predicted to a very good extent from simple statistics of integrated profile and DM-SNR curve.

VI. Avenues for further research

Support vector performs quite well for the given dataset. Doing some kind of feature preprocessing like PCA, to extract important features before inputting it to the model might help. Extracting more statistics while preparing the dataset itself will help improve model performance.

References

[1] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2nd ed., 2019.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.

[5] "Support vector machine." https://en.wikipedia.org/wiki/Support_vector_machine.