Assignment-1: A mathematical essay on Linear Regression

Vasudev Gupta
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai, India
7vasudevgupta@gmail.com

***Abstract*—In this essay, we will be presenting a mathematical study on linear regression models. We will be applying the linear regression model to predict the mortality rate and will try to analyse if fitting the linear regression model was the right decision. We will also try to do exploratory analysis on the data and do feature selection to feed the right data to the model. Finally, we will try to analyse if our model can be presented to the client by analysing the generalization capacity of the model on unseen data.**

I. Introduction

Machine learning techniques are quite popular these days and is helping several companies in solving quite hard problems based on data-driven approaches. Machine learning models try to approximate the function on the given training data by capturing the underlying distribution of the training data and then predicts on unseen data using the learnt distribution. While preparing the model, one assumes that training data is similar to unseen data and hence the model can generalize well to the future data.

In this essay, we will be focusing on the linear regression model. Linear models work under the assumption that residual error is normally distributed and a line can be fitted on the training data. We will discuss a detailed mathematical study in the next section.

In this essay, we are provided with the data of several regions and we need to present a study on whether low-income groups are more prone to fatality because of cancer. We are given several features based on which we can make the predictions but we need to analyse if some particular feature is affecting the model's performance significantly.

In this essay, we will be deriving solutions to linear regression models mathematically and will use existing Python libraries to fit the linear model on the given data. We will also discuss the insights we have gained from data and how we decided to choose certain features. Finally, we will see if our predictions follow the assumption we will make before fitting the model to data.

II. Linear Regression

In linear regression, model's response is approximated as a linear function of the inputs:

$$y(x) = w^T x + \epsilon = \sum_{i=1}^{D} w_i x_i + \epsilon$$

where $w^T x$ represents the inner or scalar product between the input vector x and the model's weight vector w, and ϵ is the residual error between our linear predictions and the true response.

We often assume that ϵ has a Gaussian distribution. We denote this by $N_\epsilon(\mu, \sigma^2)$, where μ is the mean and σ^2 is the variance. Mathematically, we can say:

$$p(y|x, \theta) = N(y|\mu(x), \sigma^2(x))$$

In the simplest case, we assume μ is a linear function of x, so $\mu = w^T x$, and that the noise is fixed, $\sigma^2(x) = \sigma^2$. In this case, $\theta = (w, \sigma^2)$ are the parameters of the model.

$$\mu(x) = w_0 + w_1 x = w^T x$$

where w_0 is the bias term, w_1 is the slope, and where we have defined the vector $x = (1, x)$. If w_1 is positive, it means we expect the output to increase as the input increases.

Linear regression can be made to model non-linear relationships by replacing x with some non-linear function of the inputs, $\phi(x)$. That is, we use

$$p(y|x, \theta) = N(y|w^T \phi(x), \sigma^2)$$

A. Maximum likelihood estimation

A common way to estimate the parameters of a statistical model is to compute the MLE, which is defined as

$$\theta = \arg max_\theta \log p(D|\theta)$$

It is common to assume the training examples are independent and identically distributed. This means we can write the log-likelihood as follows:

$$l(\theta) = \log p(D|\theta) = \sum_{i=1}^{N} \log p(y_i|x_i, \theta)$$

Instead of maximizing the log-likelihood, we can equivalently minimize the negative log likelihood:

$$NLL(\theta) = -\sum_{i=1}^{N} logp(y_i|x_i, \theta)$$

Now, if we apply the method of MLE to the linear regression and insert the definition of the Gaussian into the above, we find that the log likelihood is given by

$$l(\theta) = \sum_{i=1}^{N} \log[(\frac{1}{2\pi\sigma^2})^{\frac{1}{2}} \exp^{(-\frac{1}{2\sigma^2}(y_i - w^T x_i)^2)}]$$

$$= -\frac{1}{2\sigma^2} RSS(w) - \frac{N}{2} log(2\pi\sigma^2)$$

where RSS stands for **residual sum of squares** and is defined by

$$RSS(w) = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

It can also be written as the square of the l2 norm of the vector of residual errors:

$$RSS(w) = ||\epsilon||_2^2 = \sum_{i=1}^{N} \epsilon_i^2$$

where $\epsilon_i = (y_i - w^T x_i)$.

B. Derivation of the MLE

First, we rewrite the objective in a form that is more amenable to differentiation:

$$NLL(w) = \frac{1}{2}(y - Xw)^T(y - Xw) = \frac{1}{2} w^T (X^T X) w - w^T (X^T y)$$

where

$$X^T X = \sum_{i=1}^{N} x_i x_i^T = \sum_{i=1}^{N} \begin{pmatrix} x_{i,1}^2 & . & . & . & x_{i,1} x_{i,D} \\ & . & & & \\ & & . & & \\ & & & . & \\ x_{i,D} x_{i,1} & . & . & . & x_{i,D}^2 \end{pmatrix}$$

is the sum of squares matrix and

$$X^T y = \sum_{i=1}^{N} x_i y_i.$$

Using results from above equation, we see that the gradient of this is given by

$$g(w) = [X^T X w - X^T y] = \sum_{i=1}^{N} x_i (w^T x_i - y_i)$$

equating to zero we get,

$$X^T X w = X^T y$$

This is known as the normal equation. The corresponding solution wˆ to this linear system of equations is called the ordinary least squares or OLS solution, which is given by

$$w_{OLS} = (X^T X)^{-1} X^T y$$

III. The Problem

In subsequent sections, we will be analysing and cleaning the data provided by the client. We will also fit the machine learning model to the data and will inspect if model assumptions meet its application on real-world data.

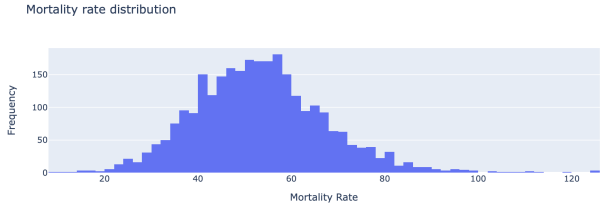
A. Data Analysis

As we discussed in the previous section that linear regression works based on assumption that the conditional distribution of the target variable is normal. Hence to verify if it's a good idea to implement linear regression to this problem, we plotted the histogram of the target variable (i.e. Mortality Rate).

```
mortality_rate = data.loc[:, "Mortality_Rate"]
print({"min": mortality_rate.min(), "max": mortality_rate.max()})

fig = go.Figure()
fig.add_trace(go.Histogram(x=mortality_rate))
fig.update_layout(title='Mortality rate distribution', yaxis={'title': "Frequency"}, xaxis={'title': "Mortality Rate"})
fig.show()
```

```
{'min': 9.2, 'max': 125.6}
```



The histogram of mortality rate looked like a normal curve for most of the samples, hence we decided to approximate the mortality rate with the linear model in the rest of the essay.

Now, we decided to clean the data as several features had missing values while few features had string values. Machine learning models can understand only numbers, hence we encoded string features into numbers and called them as categorical variables. We also dropped the samples which had many missing values and imputed the few features with mean of that feature.

For example, column recent_trend had random values in several samples, hence we decided to combine all those samples and imputed them with 0.

```
data["recent_trend"].value_counts()
```

```
stable     2382
falling     197
-           151
rising       39
*            28
__           12
Name: recent_trend, dtype: int64
```

```
data['less_than_16'] = data["recent_trend"].map(lambda x: 1 if x in ["*", "stable"] else 0)
data['rising'] = data["recent_trend"].map(lambda x: 1 if x == "rising" else 0)
data['falling'] = data["recent_trend"].map(lambda x: 1 if x == "falling" else 0)
data.drop(["recent_trend"], inplace=True, axis=1)
data.head()
```

We applied similar procedure to most of the features and obtained the cleaned version of the data. Now, we decided to analyse data further and select all the relevant features out of the bulk of features. We will discuss more on that in the next section.

B. Data Analysis

It is well known said in Machine Learning that "feeding junk in, get the junk out". Keeping this in mind, we tried to analyse and find out all the important features from our

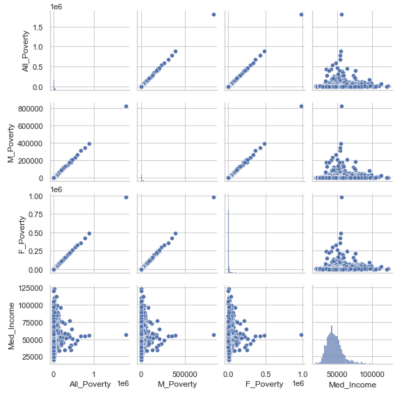
cleaned version of the data and decided to train our model only on that data.

To find out important features, we first discarded all the features that are highly correlated since highly correlated features will make our model unnecessarily complex and the model may get more prone to over-fitting to those particular features.

In the following graph, we tried to plot a scatter plot between several features and tried to visualize if a pair of features follow similar trend.

```
columns = ["All_Poverty", "M_Poverty", "F_Poverty", "Med_Income"]

sns.set(style='whitegrid', context='notebook')
sns.pairplot(X.loc[:, columns], height=2)
plt.show()
```



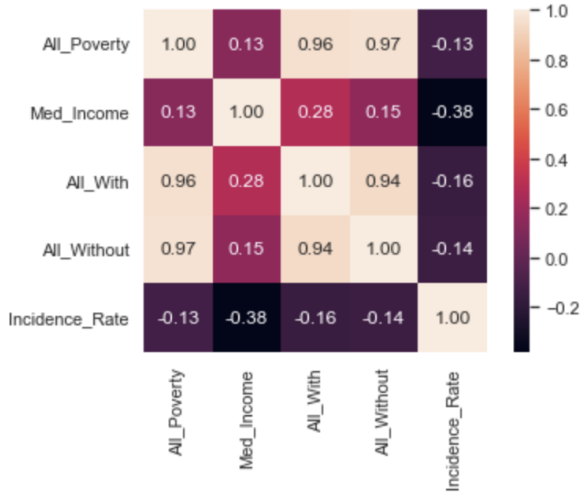
Based on the above graph, we concluded that ‘M_Poverty‘ and ‘F_Poverty‘ are highly correlated to ‘All_Poverty‘. Hence we decided to drop these features.

Further to get show correlation mathematically, we plotted a heat map of correlation among several features in the following figure.

```
info = np.corrcoef(X[columns].values.T)
sns.heatmap(info, cbar=True, annot=True, square=True,
plt.show()
```

We did a similar thing with another subset of features and realized that ‘All_With‘ and ‘All_Without‘ can be dropped further as they are highly correlated with ‘All_Poverty‘.

```
cm = np.corrcoef(X[columns].values.T)
sns.heatmap(cm, cbar=True, annot=True, square=True, fm
plt.show()
```



C. *Model Fitting*

After selecting the subset of features from complete data, we decided to use ‘statsmodels‘ library for fitting the linear model over the training data. We got an R-squared score of 0.77. After seeing the coefficients of the model, we realized that several features had coefficients close to 0. Hence we decided to train our next model on a subset of features whose coefficients had significant values.

```
model = OLS(y, X, hasconst=True)
result = model.fit()

result.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.770
Model:	OLS	**Adj. R-squared:**	0.769
Method:	Least Squares	**F-statistic:**	880.9
Date:	Fri, 17 Sep 2021	**Prob (F-statistic):**	0.00
Time:	15:08:28	**Log-Likelihood:**	-8797.2
No. Observations:	2641	**AIC:**	1.762e+04
Df Residuals:	2630	**BIC:**	1.768e+04
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
All_Poverty	-2.872e-05	1.5e-05	-1.916	0.055	-5.81e-05	6.69e-07
Med_Income	-0.0002	1.35e-05	-11.205	0.000	-0.000	-0.000
M_With	-7.576e-06	3.87e-05	-0.196	0.845	-8.34e-05	6.82e-05
M_Without	0.0002	9.86e-05	1.637	0.102	-3.19e-05	0.000
F_With	1.284e-05	3.88e-05	0.331	0.741	-6.33e-05	8.9e-05
F_Without	-0.0001	9.79e-05	-1.415	0.157	-0.000	5.34e-05
Incidence_Rate	0.6519	0.008	78.702	0.000	0.636	0.668
Avg_Ann_Incidence	-0.0063	0.003	-2.125	0.034	-0.012	-0.000
less_than_16	3.9987	0.401	9.973	0.000	3.212	4.785
falling	4.9561	0.541	9.160	0.000	3.895	6.017
rising	2.2869	0.856	2.670	0.008	0.608	3.966

We iteratively did this i.e. removed 1 feature and visualize if it’s affecting the model’s performance by a significant amount. Based on this approach, we were able to remove 5 features without compromising the model’s performance. The

significance of all the final features is shown in the following graph:

Dep. Variable:	y	**R-squared:**	0.769
Model:	OLS	**Adj. R-squared:**	0.769
Method:	Least Squares	**F-statistic:**	1759.
Date:	Fri, 17 Sep 2021	**Prob (F-statistic):**	0.00
Time:	15:08:36	**Log-Likelihood:**	-8801.0
No. Observations:	2641	**AIC:**	1.761e+04
Df Residuals:	2635	**BIC:**	1.765e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Med_Income	-0.0001	1.18e-05	-12.131	0.000	-0.000	-0.000
Incidence_Rate	0.6512	0.008	80.657	0.000	0.635	0.667
Avg_Ann_Incidence	-0.0036	0.001	-4.574	0.000	-0.005	-0.002
less_than_16	3.8852	0.390	9.963	0.000	3.121	4.650
falling	4.8445	0.536	9.044	0.000	3.794	5.895
rising	2.2007	0.852	2.583	0.010	0.530	3.871
constant	10.9304	0.749	14.588	0.000	9.461	12.400

Omnibus:	341.383	**Durbin-Watson:**	1.724
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3575.474
Skew:	-0.180	**Prob(JB):**	0.00
Kurtosis:	8.689	**Cond. No.**	2.07e+20

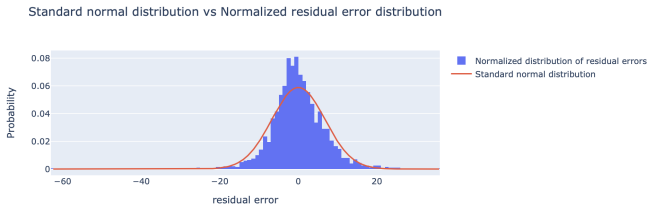
D. Model's Analysis

Further, we tried to analyse the model's predictions and tried to inspect if the model's results are as per the assumptions we made before fitting the model.

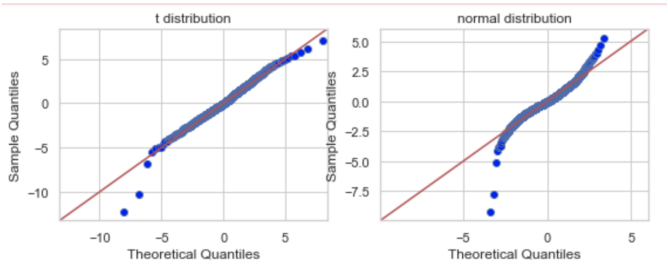
We plotted the distribution of the residuals. Note that we assumed that the linear model works based on assumption that residual error distribution is normal. We showed that this assumption holds with our model too.

```
mu, sigma = np.mean(residual), np.std(residual)
pdf = stats.norm.pdf(sorted(residual), mu, sigma)

fig = go.Figure()
fig.add_trace(go.Histogram(x=residual, name="Normalized distribution of residual errors", histnorm='probability'))
fig.add_trace(go.Scatter(x=sorted(residual), y=pdf, name="Standard normal distribution", mode="lines"))
fig.update_layout(title='Standard normal distribution vs Normalized residual error distribution', yaxis={'title': 'Probab
fig.show()
```

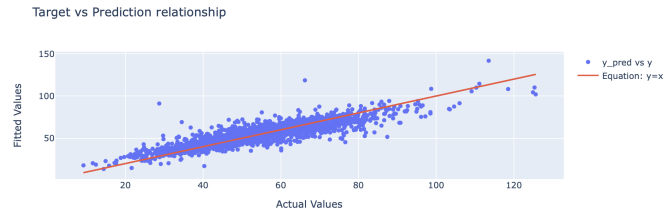


We also plotted the Q-Q plots to confirm if the residuals adhere more closely to the t-than normal distribution (fatter tails).



Finally, we plotted a graph between the model's prediction and target to visualize if predictions are following the trend of the target variable.

```
fig= go.Figure()
fig.add_trace(go.Scatter(x=y, y=y_pred, name='y_pred vs y', mode="markers"))
fig.add_trace(go.Scatter(x=y, y=y, name='Equation: y=x', mode="lines"))
fig.update_layout(title= 'Target vs Prediction relationship', yaxis={'title': 'Fitted Values'}, xaxis={'title': 'Actual
```



IV. Improvements

Instead of removing features with a high level of collinearity, we will use regularized models. Regularized models address collinearity by shrinking coefficients towards 0, thus reducing model variance.

A. Ridge Regression

We modify our objective function by introducing an $\ell 2$ *norm* penalty to penalise larger coefficients.

$$\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{t}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

where λ is the Regularization strength. Regularization improves the conditioning of the problem and reduces the variance of the estimates.
Post hyper-parameter tuning, we get λ to be,

$$\lambda = 10^6$$

Results:

R^2 statistic: **0.21338**

Variable	Coefficients
All_Poverty	-7.99×10^{-5}
Med_Income	-5.41×10^{-4}
All_With	1.46×10^{-5}
All_Without	-1.05×10^{-5}
falling	1.18×10^{-4}
rising	-1.13×10^{-4}
Constant	78.58

B. Lasso Regression

We modify our objective function by introducing an $\ell 1\ norm$ penalty to penalise larger coefficients. The Lasso is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent

$$\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{t}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

where λ is the Regularization strength.
Post hyper-parameter tuning, we get λ to be,

$$\lambda = 10^3$$

Results:

R^2 statistic: **0.21326**

Variable	Coefficients
All_Poverty	-7.12×10^{-5}
Med_Income	-5.40×10^{-4}
All_With	1.46×10^{-5}
All_Without	-1.04×10^{-5}
falling	0
rising	0
Constant	77.99

C. ElasticNet Regression

ElasticNet is a linear regression model trained with both ℓ_1 and ℓ_2 norm regularization of the coefficients. This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge.

$$\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{t}\|_2^2 + \lambda\rho\|\mathbf{w}\|_1 + \frac{\lambda(1-\rho)}{2}\|\mathbf{w}\|_2^2$$

where λ is the Regularization strength and ρ is the $\ell_1\ ratio$ which controls the convex combination of ℓ_1 and ℓ_2 norm penalties.
Post hyper-parameter tuning, we get λ to be,

$$\lambda = 10^3$$

$$\rho = 0.1$$

Results:

R^2 statistic: **0.21338**

Variable	Coefficients
All_Poverty	-7.98×10^{-5}
Med_Income	-5.40×10^{-4}
All_With	1.46×10^{-5}
All_Without	-1.04×10^{-5}
falling	0
rising	0
Constant	78.57

The ElasticNet Regression model gives us a score equal to that of the Ridge Regression model while being sparser at the same time. In accordance with Occam's Razor principle, we will choose the **ElasticNet Regression model to be our final model**.

V. Conclusions

In this essay, we derived the solution of linear regression mathematically based on some assumptions. We also applied Linear regression techniques to real-world data and tried to inspect if the fitted model follows the assumptions. We conclude that a simple model like linear regression can be applied to real-world complex datasets if data follows the underlying assumptions.

References

[1] Christopher M. Bishop, "Pattern Recognition and Machine Learning"
[2] Kevin P. Murphy, "Machine Learning, A Probabilistic Perspective"
[3] Yunus Koloğlu, Hasan Birinci, Sevde Ilgaz Kanalmaz, Burhan Özyılmaz, "A Multiple Linear Regression Approach For Estimating the Market Value of Football Players in Forward Position"

Assignment-2: A mathematical essay on Logistic Regression

Vasudev Gupta
Interdisciplinary Dual Degree - Data Science
Indian Institute of Technology Madras
Chennai, India
me18b182@smail.iitm.ac.in

***Abstract*—We will present a mathematical research on logistic regression models in this paper. We will use the logistic regression model to estimate the survival rate and will attempt to determine whether using the logistic regression model was the correct decision. We will also attempt exploratory data analysis and feature selection in order to give the correct data to the model. Finally, we'll check if our model can generalise effectively to unseen data.**

I. Introduction

Machine learning techniques are becoming increasingly popular, and they are assisting a number of organisations in tackling difficult challenges using data-driven approaches. Machine learning methods attempt to estimate the function on provided training data by capturing the underlying distribution of the training data and then predict on unseen data using the learned distribution. When building the model, one expects that the training data is comparable to previously unseen data and that the model will generalise well to future data.

The logistic regression model will be the focus of this essay. Logistic regression works on the assumption that the conditional distribution of the predicted variable with regard to the input data belongs of the Bernoulli distribution family. In the next part, we will go over a comprehensive mathematical analysis.

In this essay, we are given data on the Titanic disaster, and we must analyse which groups of passengers are more likely to survive and which groups of passengers died in the disaster. We are provided numerous attributes on which to base our predictions, but we must determine whether any of them can have a substantial impact on the model's performance.

In this article, we will deduce mathematical solutions to logistic regression models and use existing Python libraries to fit the logistic model to the provided data. We'll also talk about the data insights we've gleaned and why we opted to change or remove specific features.

II. Logistic Regression

Logistic regression is an extension of linear regression, and by making two adjustments, we may expand linear regression to the (binary) classification scenario. First, we substitute the Gaussian distribution for y with a Bernoulli distribution, which is better suited for binary responses, $y \in \{0, 1\}$. That is, we employ

$$p(y|x, w) = Ber(y|\mu(x))$$

where $\mu(x) = E[y|x] = p(y = 1|x)$. Second, we compute a linear combination of the inputs, as before, but then we pass this through a function that ensures $0 \leq \mu(x) \leq 1$ by defining

$$\mu(x) = sigm(w^T x)$$

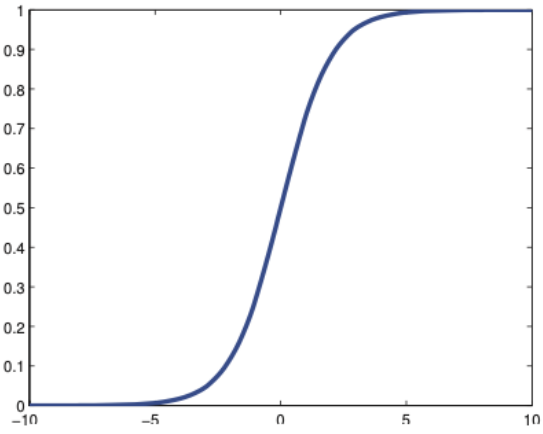
where $sigm(\eta)$ refers to the sigmoid function, also known as the logistic function. This is defined as

$$sigm(\eta) = \frac{\exp(\eta)}{\exp(\eta) + 1}$$

Putting these two steps together we get

$$p(y|x, w) = Ber(y|sigm(w^T x))$$

This is called logistic regression due to its similarity to linear regression (although it is a form of classification).



A simple example of logistic regression is shown in above figure, where we plot

$$p(y_i = 1|x_i, w) = sigm(w_0 + w_1 x_i)$$

If we threshold the output probability at 0.5, we can induce a decision rule of the form

$$\hat{y}(x) = 1 \iff p(y = 1|x) > 0.5$$

Logistic regression may be readily expanded to handle higher-dimensional inputs. If we set the threshold for these probabilities to 0.5, we have a linear decision boundary with the normal (perpendicular) provided by w. In the next part, we shall derive the loss function associated with logistic regression.

A. Maximum likelihood estimation

The negative log-likelihood for logistic regression is given by

$$NLL(w) = -\sum_{i=0}^{N} \log[\mu_i^{I(y_i=1)} \times (1-\mu_i)^{I(y_i=0)}]$$

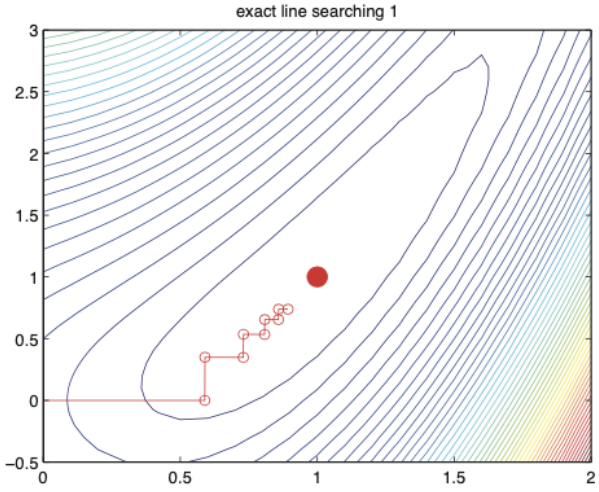
$$= -\sum_{i=0}^{N} [y_i \log \mu_i + (1-y_i)\log(1-\mu_i)]$$

This is also called the cross-entropy error function. Unlike linear regression, we can no longer write down the Maximum Likelihood Estimation in closed form. Instead, we need to use an optimization algorithm to compute it. For this, we need to derive the gradient and Hessian. In the case of logistic regression, the gradient and Hessian are given by the following

$$g = \frac{d}{dw} f(w) = \sum_i (\mu_i - y_i) x_i = X^T(\mu - y)$$

$$H = \frac{d}{dw} g(w)^T = \sum_i (\Delta_w \mu_i) x_i^T = X^T S X$$

where $S = diag(\mu_i(1-\mu_i))$. Here the NLL is convex and has a unique global minimum.



The simplest algorithm for unconstrained optimization is gradient descent (as shown in above figure), also known as steepest descent. This can be written as follows:

$$\Theta_{k+1} = \Theta_k - \eta_k g_k$$

where η_k is the learning rate.

III. The Problem

In the following sections, we will analyse and clean the Titanic dataset. We will also fit the machine learning model to the data and test if the model can make accurate predictions on previously unknown data.

A. Data Analysis and Feature Engineering

There were 10 feature columns and 1 target column in the given data. Given ten input variables, we must predict the target variable.

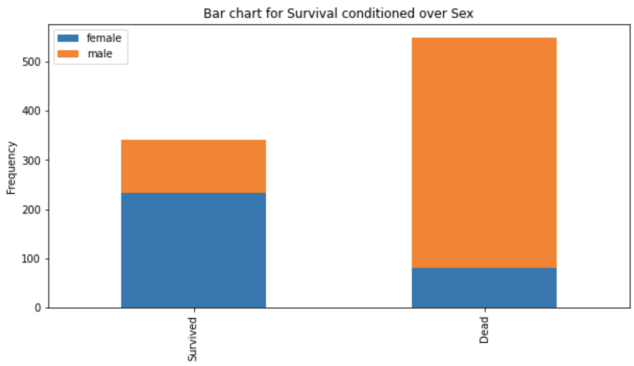
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

We imputed the columns with the median of their respective columns because some of the samples had `NaN` values.

We analysed the data by classifying characteristics according to the target variable (i.e. `Survival`) and displaying bar graphs to evaluate the importance of various attributes in predicting survival rate.

We plotted bar chart based of the Gender in the following figure:



We found that more percentage of females survived compared to the male passengers.

Further, we tried to categorize the passengers based on their age group and marital status. For extracting this information

from the data, we relied on `Age` and `Name` columns. We used `Name` column by extracting prefix and further dividing it's classes into `Mr`, `Mrs`, `Miss`, `Master` and `Other` category.

We calculated the probability of survival based on this column in the next figure:

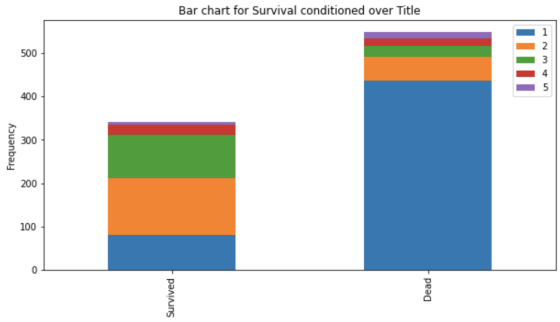
```
for dataset in [train, test]:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Ca
    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

train[['Title', 'Survived']].groupby(['Title'], as_index=False).mean()
```

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Other	0.347826

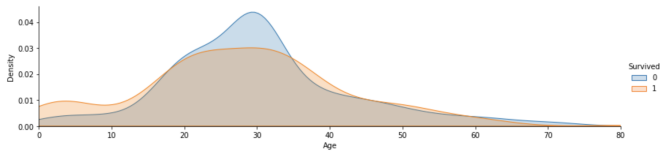
```
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Other": 5}
for dataset in [train, test]:
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

bar_chart('Title')
```



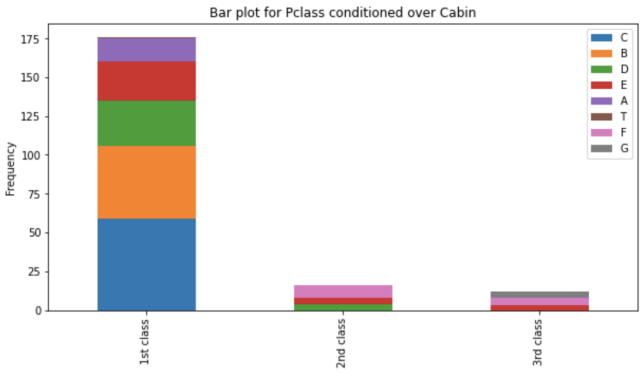
We concluded that married females outlived unmarried females and that married females outlived married males by a large margin. We hypothesised that married men died protecting their families.

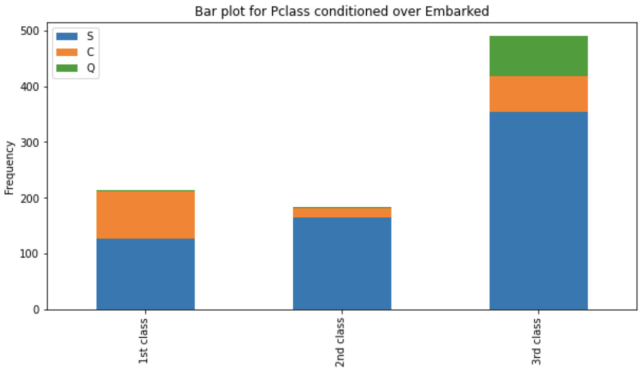
For understanding the age group of the survived people, we plotted the density plots of `Age` conditioned over `Survival`.



We discovered that persons between the ages of 18 and 35 died at a higher rate than people of other ages.

Finally, we attempted to comprehend the class of individuals by visualising which class of people dwelt in which cabin and where they embarked.





B. Model Fitting

We trained a Logistic regression model on the data using the `sklearn` library and achieved accuracy of `80%`

We have created a confusion matrix to better comprehend the ratio of `false negatives` and `false positives`.

```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(target, model.predict(train_data))

index = ["Prediction positive", "Prediction negative"]
columns = ["Label positive", "Label negative"]
pd.DataFrame(matrix, columns=columns, index=index)
```

	Label positive	Label negative
Prediction positive	469	80
Prediction negative	94	248

IV. Improvements

A. Feature Scaling

We **standardize** numerical features to improve the model's convergence. Standardization is done as follows,

$$x_{std} = \frac{x - \mu}{\sigma}$$

B. Feature Selection

We use *sklearn's* inbuilt filter-based feature selection tool, **SelectKBest**. It is a uni-variate feature selection tool which selects the k highest scoring features. We score the features on the estimate of mutual information **(MI)** between the feature and the discrete target variable.

C. Classification Model

We will use **Logistic Regression** as our classification model, as this is the subject of this paper.

Pipeline - Grid Search

This pipeline model has several parameters which we can tune to obtain better results, let's perform Grid Search to find out the optimal parameters. In order to perform Grid Search, we need to specify the set of parameters over which the search is to be done. Let's have a look at the parameter grid.

Parameters	Values
k	[2, 3, ... , 8]
C	[0.1, 1, 10, 100]
penalty	['l1', 'l2']
class_weight	['balanced', None]

On completing grid search, the following set of parameters give best performance.

Parameters	Optimal Values
k	4
C	1.0
penalty	'l2'
class_weight	'balanced'

Results: Used stratified K-fold (K=5) cross-validation to evaluate the model's f1 score using the above found optimal set of parameters.

$$\boxed{Accuracy\ Score\ =\ \mathbf{0.808};\ F1\ Score\ =\mathbf{0.762}}$$

Although accuracy decreased by a small margin, we see a slight improvement in the **F1 Score** of the model. As the dataset is imbalanced, the F1 Score is a better measure of the model's performance. Further we've reduced the number of features by half, thereby reducing the model's complexity while improving it's performance.

V. Conclusions

In this essay, we derived the solution of logistic regression mathematically based on some assumptions. We also applied Logistic regression techniques to real-world data and tried to inspect if the fitted model can be predict well. We conclude that provided the underlying assumptions are met, a basic model like logistic regression may be used on real-world complicated datasets.

References

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

[2] Kevin P. Murphy, "Machine Learning, A Probabilistic Perspective"

[3] Yunus Koloğlu, Hasan Birinci, Sevde Ilgaz Kanalmaz, Burhan Özyılmaz, "A Multiple Linear Regression Approach For Estimating the Market Value of Football Players in Forward Position"

Assignment-3: A mathematical essay on Naive Bayes Classifier

Vasudev Gupta
Interdisciplinary Dual Degree - Data Science
Indian Institute of Technology Madras
Chennai, India
me18b182@smail.iitm.ac.in

***Abstract*—In this work, we will give a mathematical study of the Naive Bayes Classifier. We will use the Naive Bayes Classifier to determine whether the person earns more than $50K per year. In order to provide the correct data to the model, we will also try exploratory data analysis and feature selection. Finally, we will examine if our model can properly generalise to previously unknown data.**

I. Introduction

Machine learning techniques are gaining popularity, helping many organisations solve challenging issues using data-driven approaches. Machine learning approaches try to estimate the function based on supplied training data by capturing the underlying distribution of the training data and then predicting unseen data using the learnt distribution. When creating the model, one wants the training data to be similar to previously unseen data and generalise well to future data.

This essay will concentrate on the Naive Bayes Classifier. The Naive Bayes Classifier is predicated on the assumption that the conditional distribution of the predicted variable concerning the input data is of the Gaussian distribution family and that all variables are conditionally independent given the class label. In the following section, we will go over a thorough mathematical analysis.

Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics) provide us with data from the 1994 Census Bureau database in this article. The main objective is to establish whether a person earns more than $50,000 per year. We are given a plethora of qualities to base our predictions on, but we must evaluate if any of them may significantly influence the model's performance.

This article will infer mathematical solutions to the Naive Bayes Classifier and use existing Python packages to fit the Bayes Classifier to the data supplied. We will also discuss the data insights we discovered and why we chose to alter or delete particular features.

II. Naive Bayes Classifier

We will assume that conditional distribution given the class belongs to the Gaussian distribution.

$$X|y=+1 = N(\mu_+, I); P(y=+1) = a$$

$$X|y=-1 = N(\mu_-, I); P(y=-1) = 1-a$$

Note: Above, we are assuming co-variance matrix to be **I** which means that our features are conditionally independent of each other.

Applying Bayes rule, we get

$$P(Y=1|X=x) = \frac{f_{X|y}(x|+1)P(y=+1)}{f_{X|y}(x|+1)P(y=+1) + f_{X|y}(x|-1)P(y=-1)}$$

$$= \frac{a\exp\frac{-1}{2}||x-\mu_+||^2}{a\exp\frac{-1}{2}||x-\mu_+||^2 + (1-a)\exp\frac{-1}{2}||x-\mu_-||^2}$$

On simplifying, we get

$$= \frac{1}{1+\exp -(w^T X + b)}$$

where,

$$w = \mu_+ - \mu_-$$

$$b = \frac{-1}{2}||\mu_+||^2 + \frac{1}{2}||\mu_-||^2 + \log\frac{a}{1-a}$$

Probability over complete data can be found by multiplying individual probabilities. We get the following equation:

$$p(x|y=c,\theta) = \prod_{j=1}^{D} p(x_j|y=c,\theta_{jc})$$

Now, for training this model, we can estimate the Maximum Likelihood estimate. After maximising the log of likelihood, we get the solution to above equation as following:

$$\mu_1^{ML} = \frac{1}{N_1}\sum_{n=1}^{N}\mathbf{1}_{y_n=c_1}x_n$$

$$\mu_2^{ML} = \frac{1}{N_2}\sum_{n=1}^{N}\mathbf{1}_{y_n=c_2}x_n$$

$$a = \frac{1}{N_1}\sum_{n=1}^{N}\mathbf{1}_{y_n=c_1} = \frac{N_1}{N_1+N_2}$$

III. The problem

Given the data from the 1994 Census bureau database by Ronny Kohavi and Barry Becker, the key task is to predict whether a person is making over $\$50K$ annually. We will first analyse the data before attempting to fit the Naive Bayes model to it. The trained model will next be tested on an unknown dataset.

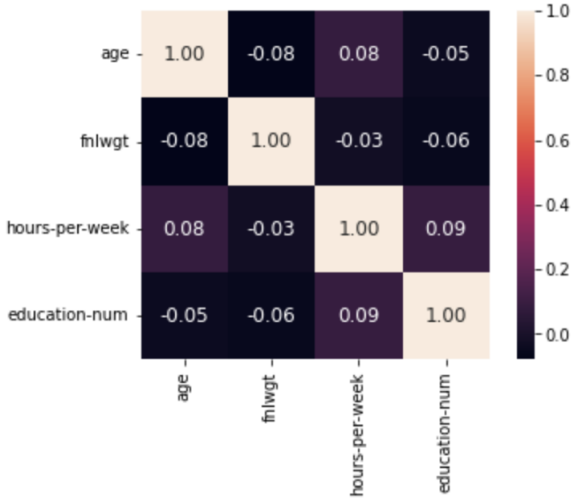
A. Data Analysis and Feature Engineering

We observed several columns in the dataset that has `?` as values, hence we decided to replace these values with `NaN` values and then tried to analyse features with missing values using following table:

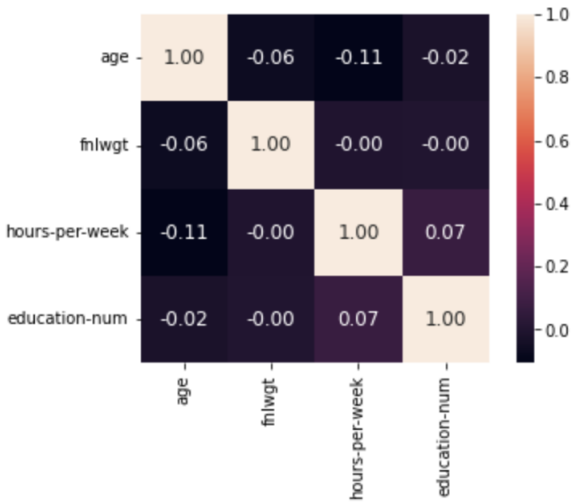
```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32560 non-null  int64
 1   workclass       30724 non-null  object
 2   fnlwgt          32560 non-null  int64
 3   education       32560 non-null  object
 4   education-num   32560 non-null  int64
 5   marital-status  32560 non-null  object
 6   occupation      30717 non-null  object
 7   relationship    32560 non-null  object
 8   race            32560 non-null  object
 9   sex             32560 non-null  object
 10  capital-gain    32560 non-null  int64
 11  capital-loss    32560 non-null  int64
 12  hours-per-week  32560 non-null  int64
 13  native-country  31977 non-null  object
 14  income          32560 non-null  object
```

Since there were very few samples with missing values, we decided to drop all those samples from the data.

While deriving the Naive Bayes classifier, we assumed that all the features are conditionally independent given the class label. We tried to find the correlation among several numerical features in the data in the following figure:



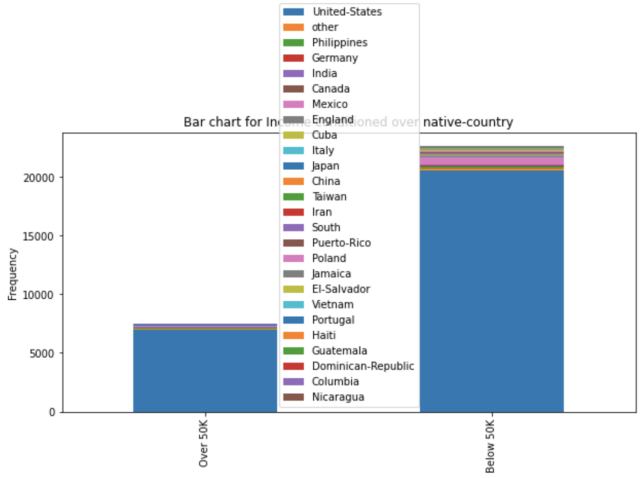
Above heat map is obtained after conditioning features on class corresponding to income less than $\$50K$.



Above heat map is obtained after conditionaling features on class corresponding to income greater than $\$50K$.

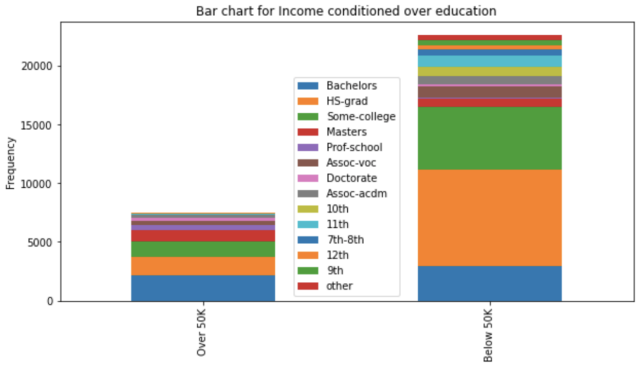
We can see that correlation among features is very low in both the figures; hence Naive Bayes could be an excellent fit for this data.

Further, before fitting the model to the data, we tried to understand the categorical features in the data using the bar plots.

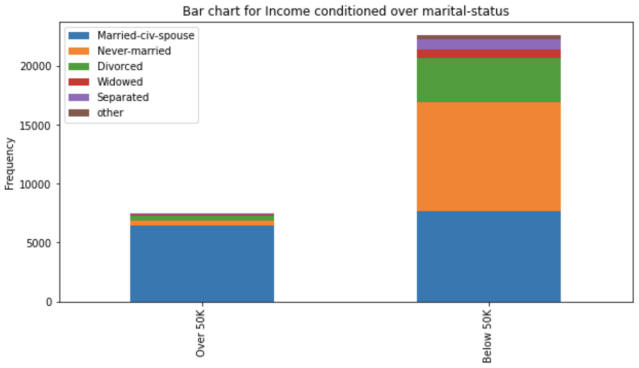


In the above plot, we can visualise that income of people is more than $\$50K$ for more people living in the United States as compared to any other country. We hypothesised that this is happening because the cost of living is more in the United States than in any other country. Hence, if a person is living in the United States, then it is very likely that he/she is going to earn a large amount of money.

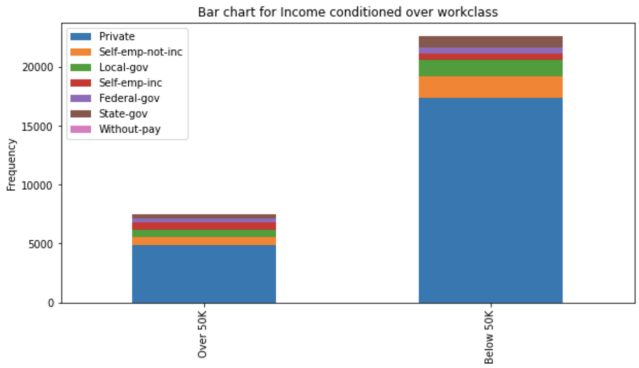
Further, we analysed whether educated people earn more money compared to school/college dropouts.



We observed that people with a higher education degree earn more compared to people who leave education early. This is expected as people who have completed higher education are more skilled and value businesses more.



We further checked if married people earn more compared to unmarried people/students. We found that married people make more money compared to other categories. We hypothesised that married people are usually of more age and hence they are more likely to earn more due to the experience gained over years of job.



Further, we tried to understand which occupations are more rewarding in terms of income. We found that people in the private sector earn more money than people from other sectors or working for govt.

B. Model Fitting

We fitted the Naive Bayes Classifier using the `sklearn` library. We analysed the model after training it on the unseen data. To ensure that test data is new to the model, we sampled the test data from the overall data (provided in `adult.csv`) before training and didn't use the test data for training.

	Predicted Positive	**Predicted Negative**
Actual Positive	4313	231
Actual Negative	1018	471

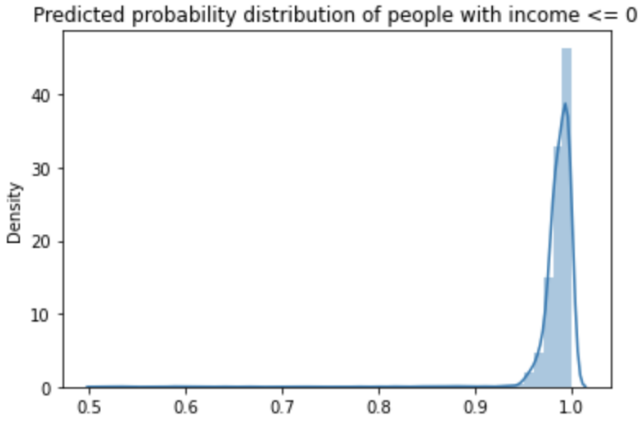
The above table shows the confusion matrix of the trained model on the unseen data. In the following table, we tried to analyse the confidence of prediction belonging to each category.

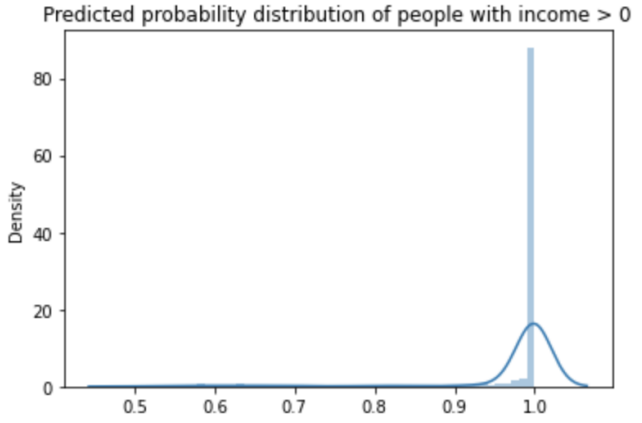
Percentage of people with predicted probability > 0.9

Income $< \$50K$	99%
Income $> \$50K$	92%

We found that the model can predict the negative class with more confidence when compared to the positive category. After further inspecting the data, we found that model assumptions do not meet data with positive category correctly. Hence, the model is not able to be utterly sure while predicting the positive category.

To further confirm our hypothesis, we plotted the probability distribution function in the following figures:





The above figures clearly show that conditional distribution over positive class violates Gaussian assumption, and hence the model is less sure of predicting this category.

IV. Improvements

We trained other classifiers on the data provided in this problem statement and compared the performance of all the models in the following table:

Classifier	**Training Accuracy**	**Validation Accuracy**
Naive Bayes	76.29	76.13
Logistic Regression	80.77	80.47
Decision Tree	97.87	80.87
Random Forest	97.89	85.18
SVM	80.33	79.81

As we can see from above table that Random Forest Classifier performed best on the given dataset.

V. Conclusions

In this essay, we derived the solution of Naive Bayes mathematically based on some assumptions. We also applied Naive Bayes techniques to real-world data and inspected whether the fitted model can predict well. We conclude that a simple model like Naive Bayes may be used on complicated real-world datasets if the underlying assumptions are met.

References

[1] Christopher M. Bishop, "Pattern Recognition and Machine Learning"
[2] Kevin P. Murphy, "Machine Learning, A Probabilistic Perspective"

Assignment-4: A mathematical essay on Decision Trees

Vasudev Gupta
Interdisciplinary Dual Degree - Data Science
Indian Institute of Technology Madras
Chennai, India
me18b182@smail.iitm.ac.in

***Abstract*—In this work, we will present a mathematical overview of the Decision Trees. We will use the Decision Trees is to classify a car based on its safety. In order to provide the correct data to the model, we will also try exploratory data analysis and feature selection. Finally, we will examine if our model can properly generalise to previously unknown data.**

I. Introduction

Machine learning techniques are gaining popularity, helping many organisations solve challenging issues using data-driven approaches. Machine learning approaches try to estimate the function based on supplied training data by capturing the underlying distribution of the training data and then predicting unseen data using the learnt distribution. When creating the model, one wants the training data to be similar to previously unseen data and generalise well to future data.

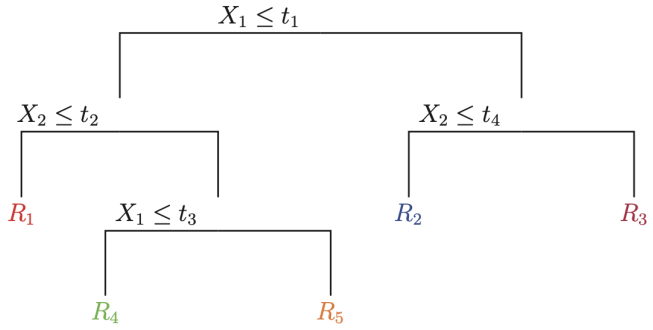
Classification and regression trees, also called decision trees, are defined by recursively partitioning the input space and defining a local model in each resulting region of input space. In the following section, we will go over a thorough mathematical analysis.

The Car Evaluation Database was derived from a simple hierarchical decision model. The prediction task is to classify a car based on its safety. We are given many qualities to base our predictions on, but we must evaluate if any of them may significantly influence the model's performance.

This article will infer a mathematical overview of the Decision Trees and use existing Python packages to fit the Decision Trees to the given data. We will also discuss the data insights we discovered and why we chose to alter or delete particular features.

II. Decision Trees

To explain the Decision trees, consider the tree in the figure below. The first node asks if x_1 is less than some threshold t_1. If yes, we then ask if x_2 is less than some other threshold t_2. If yes, we are in the bottom left quadrant of space, R_1. If no, we ask if x_1 is less than t_3. And so on.



We can write the model in the following form:

$$f(x) = E[y|x] = \sum_{m=1}^{M} w_m I(x \in R_m) = \sum_{m=1}^{M} w_m \phi(x; v_m)$$

where R_m is the m'th region, w_m is the mean response in this region, and v_m encodes the choice of variable to split on, and the threshold value, on the path from the root to the m'th leaf. This makes it clear that a decision tree is just a an adaptive basis-function model, where the basis functions define the regions, and the weights specify the response value in each region. We discuss how to find these basis functions below.

We can generalize this to the classification setting by storing the distribution over class labels in each leaf, instead of the mean response.

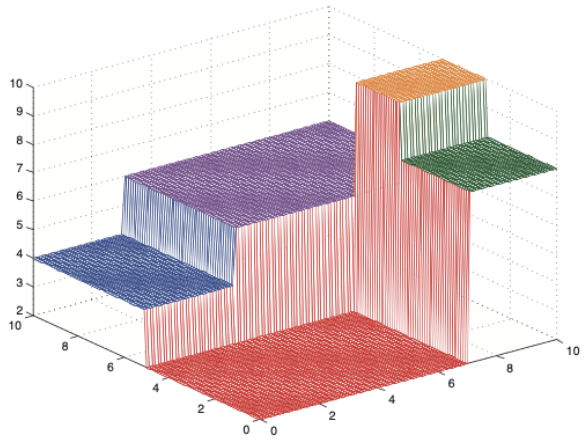
A. Growing a tree

Finding the optimal partitioning of the data is NP-complete, so it is common to use the greedy procedure shown in Algorithm 6 to compute a locally optimal MLE. The split function chooses the best feature, and the best value for that feature, as follows:

$$j^*, t^* = arg\ min\ min\ cost(xi, y_i : x_{ij} \leq t) + cost(x_i, y_i : x_{ij} > t)$$

where the cost function for a given dataset will be defined below. For notational simplicity, we have assumed all inputs are real-valued or ordinal, so it makes sense to compare a feature x_{ij} to a numeric value t. The set of possible thresholds

T_j for feature j can be obtained by sorting the unique values of x_{ij}. Although we could allow for multi-way splits (resulting in non-binary trees), this would result in data fragmentation, meaning too little data might "fall" into each subtree, resulting in overfitting.



B. Classification cost

In the classification setting, there are several ways to measure the quality of a split. First, we fit a multinoulli model to the data in the leaf satisfying the test $X_j < t$ by estimating the class-conditional probabilities as follows:

$$\pi_c = \frac{1}{|D|} \sum_{i \in D} I(y_i = c)$$

where D is the data in the leaf. Given this, there are several common error measures for evaluating a proposed partition:

1. **Misclassification rate**: We define the most probable class label as $\hat{y_c} = arg\ max_c \hat{\pi_c}$. The corresponding error rate is then

$$\frac{1}{|D|} \sum_{i \in D} I(y_i \neq \hat{y}) = 1 - \hat{\pi_{\hat{y}}}$$

2. **Entropy, or deviance:**

$$H(\hat{\pi}) = -\sum_{c=1}^{C} \hat{\pi_c} \log \hat{\pi_c}$$

III. The Problem

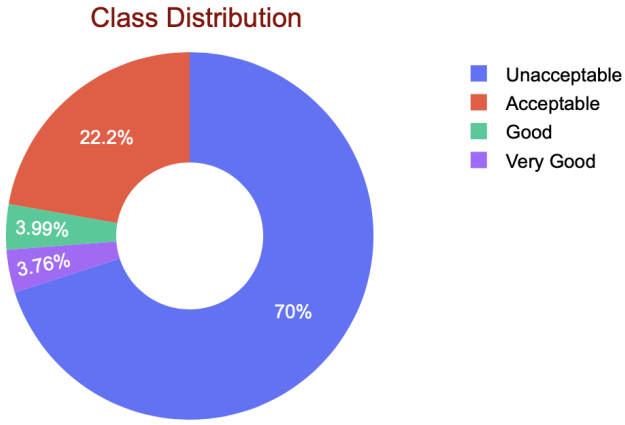
The prediction task is to classify a car based on its safety. We will first analyse the data before attempting to fit the decision trees to it. The trained model will next be tested on an unknown dataset.

A. Data Analysis and Feature Engineering

Data was clean and did not have any `NaN` values, so we did not worry about them.

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Price        1728 non-null   object
 1   Maintenance  1728 non-null   object
 2   NumDoors     1728 non-null   object
 3   NumPersons   1728 non-null   object
 4   LugBoot      1728 non-null   object
 5   Safety       1728 non-null   object
 6   Class        1728 non-null   object
```

We tried to visualize the class distribution in the data and see if there was any imbalance in the target variable.

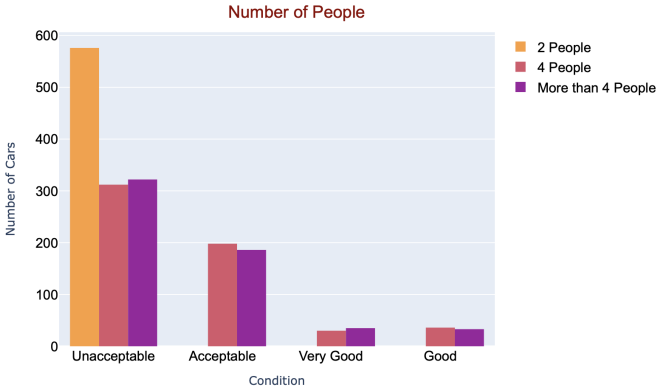


The above pie chart displays that data is quite imbalanced and most of the samples belongs to `Unacceptable` and `Acceptable` categories.

Further, before fitting the model to the data, we tried to understand the categorical features in the data using the bar plots.

The above table depicts that safety in cars is directly proportional to the number of acceptable cars. Hence, the more unsafe the car is, the more likely it will not be acceptable.

The above figure depicts that more people prefer a car with either medium/high luggage capacity.



The above figure clearly says that more people tend to buy cars which can accommodate at least four people.

From the above plot, we can conclude that highly-priced cars are unacceptable and people like to get a car for less money.

B. Model Fitting

We fitted the Decision trees using the `sklearn` library. Since tree-based models are pretty prone to overfitting, we first tuned the hyper-parameters using `RandomizedSearchCV`. We also used stratified K-fold to find the best hyper-parameters of this dataset. Accuracy score and Mathews Correlation Coefficient are presented in the following table:

Mathews Correlation Coeff	84.5%
Accuracy	93%

In following table, we report the best hyper-parameters for this model and data.

min samples split	2
min samples leaf	1
max features	”sqrt”
max depth	12
criterion	”entropy”

IV. Improvements

We selected a subset of features using the Recursive feature elimination technique. Recursive feature elimination, in short RFE, is a greedy optimization algorithm that aims to find the best performing feature subset. Each iteration trains a classification model (in this case, Decision Tree Classifier). It eliminates the worst-performing feature until all the features are exhausted or satisfy stopping criteria.

We observed that RFE helped in increasing the model's performance by significant amount. With RFE, we were able to reduce the number of features by 20%. Our new results are summarized in the following table:

Mathews Correlation Coeff	88.4%
Accuracy	94.3%

We obtained above results using the Decision Tree as a base estimator in RFE. In the base estimator, following hyper-parameters are used:

min samples split	2
max features	”auto”
max depth	12
criterion	”entropy”

V. Conclusions

In this essay, we presented the mathematical intuition behind Decision Trees. We also applied Decision Trees to real-world data and inspected whether the fitted model can predict well. We conclude that decision trees can be used on complicated real-world datasets after proper tuning.

References

[1] Christopher M. Bishop, ”Pattern Recognition and Machine Learning”
[2] Kevin P. Murphy, ”Machine Learning, A Probabilistic Perspective”

Assignment-5: A mathematical essay on Random Forest

Vasudev Gupta
Interdisciplinary Dual Degree - Data Science
Indian Institute of Technology Madras
Chennai, India
me18b182@smail.iitm.ac.in

***Abstract*—In this work, we will present a mathematical overview of the Random Forest. We will use the Random Forest to classify a car based on its safety. In order to provide the correct data to the model, we will also try exploratory data analysis and feature selection. Finally, we will examine if our model can properly generalise to previously unknown data.**

I. Introduction

Machine learning techniques are gaining popularity, helping many organisations solve challenging issues using data-driven approaches. Machine learning approaches try to estimate the function based on supplied training data by capturing the underlying distribution of the training data and then predicting unseen data using the learnt distribution. When creating the model, one wants the training data to be similar to previously unseen data and generalise well to future data.

Random Forests can be considered a specialised form of decision tree bagging. Bagging is done in such a way that at each node, just a random subset of all the attributes is considered. We will go over a full mathematical analysis in the next part.

The Car Evaluation Database was derived from a simple hierarchical decision model. The prediction task is to classify a car based on its safety. We are given many qualities to base our predictions on, but we must evaluate if any of them may significantly influence the model's performance.

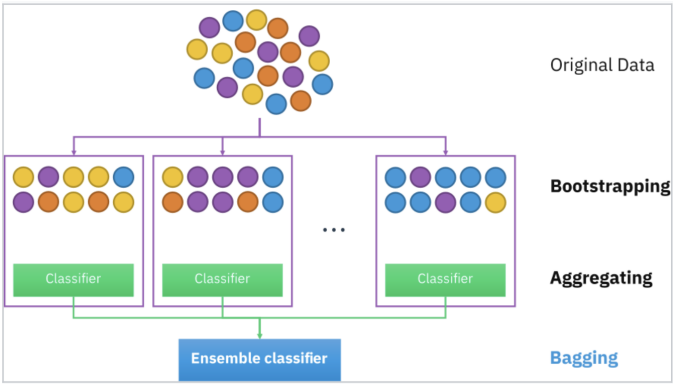
This article will infer a mathematical overview of the Random Forest and use existing Python packages to fit the Random Forest Classifier to the given data. We will also discuss the data insights we discovered and why we chose to alter or delete particular features.

II. Random Forest

Before we get into random forests, we'll go over Bagging and Decision Trees briefly.

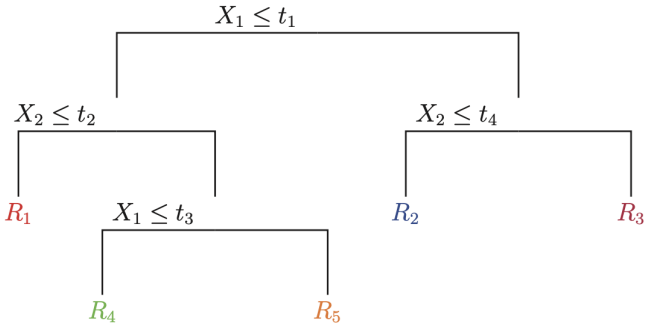
A. Bagging

Bagging is a technique for combining many distinct models developed on a subset of data simply by averaging them (or taking a majority vote). When trained on different subsets of the training data, high capacity learners produce extremely unique models.



B. Decision Trees

To explain the Decision trees, consider the tree in the figure below. The first node asks if x_1 is less than some threshold t_1. If yes, we then ask if x_2 is less than some other threshold t_2. If yes, we are in the bottom left quadrant of space, R_1. If no, we ask if x_1 is less than t_3. And so on.



We can write the model in the following form:

$$f(x) = E[y|x] = \sum_{m=1}^{M} w_m I(x \in R_m) = \sum_{m=1}^{M} w_m \phi(x; v_m)$$

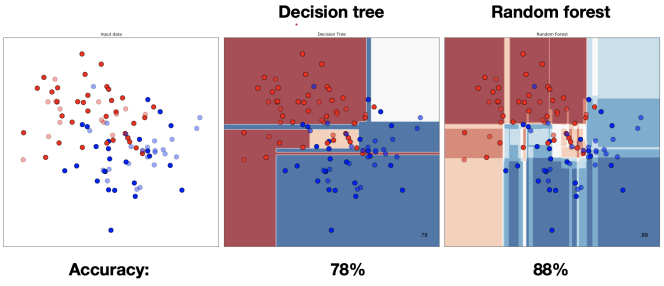
where R_m is the m'th region, w_m is the mean response in this region, and v_m encodes the choice of variable to split on, and the threshold value, on the path from the root to the m'th leaf. This makes it clear that a decision tree is just a an adaptive basis-function model, where the basis functions

define the regions, and the weights specify the response value in each region. We discuss how to find these basis functions below.

We can generalize this to the classification setting by storing the distribution over class labels in each leaf, instead of the mean response.

C. Random Forest Classifier

1) Introduction: Random Forests may be created by doing specialised bagging on decision trees. The decision trees are trained in such a way that only a random subset of all the characteristics are considered at each node. As a result, the variance of each decision tree increases during bagging. This is useful when there is a highly predictive feature, because all learners will use it and so be correlated.



On the same data, the figure above contrasts decision trees vs random forest. We can easily observe that random forest outperformed decision trees in terms of generalisation.

2) Algorithm:

- Repeat $n_{estimators}$ times:
  - Select `a*m` samples at random (with replacement preferably)
  - Learn a decision tree with the above sample, with a small variant (While constructing each node randomly select `b*d` attributes and choose the best split only among these).
- Make prediction by averaging the $n_{estimators}$ learned models.

III. The Problem

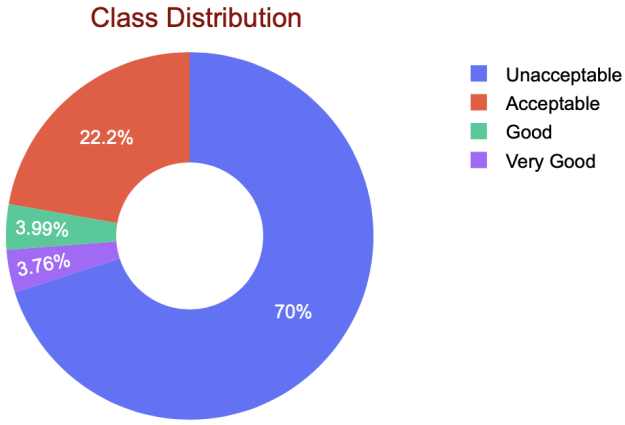
The prediction task is to classify a car based on its safety. We will first analyse the data before attempting to fit the Random Forest to it. The trained model will next be tested on an unknown dataset.

A. Data Analysis and Feature Engineering

Data was clean and did not have any `NaN` values, so we did not worry about them.

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Price        1728 non-null   object
 1   Maintenance  1728 non-null   object
 2   NumDoors     1728 non-null   object
 3   NumPersons   1728 non-null   object
 4   LugBoot      1728 non-null   object
 5   Safety       1728 non-null   object
 6   Class        1728 non-null   object
```

We tried to visualize the class distribution in the data and see if there was any imbalance in the target variable.

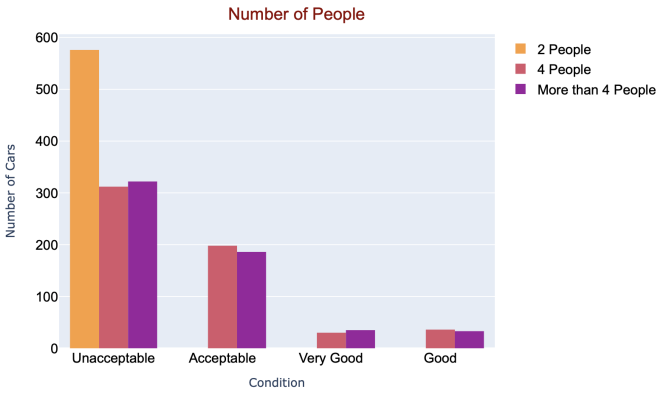


The above pie chart displays that data is quite imbalanced and most of the samples belongs to `Unacceptable` and `Acceptable` categories.

Further, before fitting the model to the data, we tried to understand the categorical features in the data using the bar plots.

The above table depicts that safety in cars is directly proportional to the number of acceptable cars. Hence, the more unsafe the car is, the more likely it will not be acceptable.

The above figure depicts that more people prefer a car with either medium/high luggage capacity.



The above figure clearly says that more people tend to buy cars which can accommodate at least four people.

From the above plot, we can conclude that highly-priced cars are unacceptable and people like to get a car for less money.

B. Model Fitting

We fitted the Random Forest using the `sklearn` library. Since tree-based models are pretty prone to overfitting, we first tuned the hyper-parameters using `RandomizedSearchCV`. We also used stratified K-fold to find the best hyper-parameters of this dataset. Accuracy score and Mathews Correlation Coefficient are presented in the following table:

Mathews Correlation Coeff	94.5%
Accuracy	97.9%

In following table, we report the best hyper-parameters for this model and data.

min samples split	2
min samples leaf	1
max features	”log2”
criterion	”gini”

IV. Improvements

We selected a subset of features using the Recursive feature elimination technique. Recursive feature elimination, in short RFE, is a greedy optimization algorithm that aims to find the best performing feature subset. Each iteration trains a classification model (in this case, Random Forest Classifier). It eliminates the worst-performing feature until all the features are exhausted or satisfy stopping criteria.

We observed that RFE helped in increasing the model's performance by significant amount. With RFE, we were able to reduce the number of features by 16%. Our new results are summarized in the following table:

Mathews Correlation Coeff	95.7%
Accuracy	98.2%

We obtained above results using the Random Forest as a base estimator in RFE. In the base estimator, following hyper-parameters are used:

min samples split	2
max depth	12
max features	”log2”
criterion	”gini”

V. Conclusions

In this essay, we presented the mathematical intuition behind Random Forest. We also applied Random Forest to real-world data and inspected whether the fitted model can predict well. We conclude that Random Forest can be used on complicated real-world datasets after proper tuning.

References

[1] Christopher M. Bishop, ”Pattern Recognition and Machine Learning”
[2] Kevin P. Murphy, ”Machine Learning, A Probabilistic Perspective”

Final Exam: Stock Market Analysis

Vasudev Gupta
Interdisciplinary Dual Degree - Data Science
Indian Institute of Technology Madras
Chennai, India
me18b182@smail.iitm.ac.in

***Abstract*—In this study, we examine equities from a variety of industries, including banking and information technology, in order to glean insights from the data. To comprehend the behaviour of the stock market, we employ specific measures and graphs from the financial world. We model stock market data sets using this information. Then we look at each stock's distinct behaviour, both individually and across sectors.**

I. Introduction

A stock market, also known as an equity market or a share market, is a gathering of buyers and sellers of stocks (also known as shares), which represent ownership claims on businesses. Stocks can be listed on a public stock exchange or privately traded, such as shares of private companies sold to investors through equity crowdfunding platforms. Stocks are classified according to the country in which the company is based.One97 Communications (Paytm) and Zomato, for example, are based in India and trade on the NSE and BSE, thus they can be regarded part of the Indian stock market, however the stocks can also be traded on other exchanges, such as as American depositary receipts (ADRs) on U.S. stock exchanges. In the 16th and 17th centuries, the first stock markets arose in Europe, mostly in port cities or trading hubs such as Antwerp, Amsterdam, and London. Due to the tiny number of corporations that issued equity, these early stock markets were more analogous to bond exchanges. In reality, because they had to be authorised by their government in order to conduct business, most early businesses were deemed semi-public. Stock exchanges, such as the New York Stock Exchange (NYSE), first appeared in America in the late 18th century, allowing for the trading of equity shares.The Buttonwood Agreement, signed by 24 New York City stockbrokers and merchants in 1792, established the NYSE. Traders and brokers would meet unofficially under a buttonwood tree on Wall Street to buy and sell shares prior to its legal incorporation. With the introduction of contemporary stock markets, a new era of regulation and professionalisation began, ensuring that buyers and sellers of stocks may trust that their transactions will be completed at acceptable prices and within a reasonable time frame. Today, there are numerous stock markets in the United States and around the world, many of which are electronically linked. As a result, markets have become more efficient and liquid.

In this study, we examine equities from a variety of industries, including banking and information technology, in order to glean insights from the data. Our work is primarily focused on analysing and comprehending the behaviour of equities, both individually and across sectors. This paper concentrates on the mathematical element of the stock market data analysis in Section II. In Section III, we focus on the given topic and conduct an in-depth investigation to comprehend stock market behaviour. Finally, Section IV summarises all of the analyses' insights and conclusions, as well as areas for further research.

II. MATHEMATICAL AND FINANCIAL TOOLS

This section explains the fundamental mathematical and financial methods used to analyse stock market data. Because investing requires a significant amount of wealth, it is critical to grasp the fundamentals before entering the battlefield.

A. Stock market data jargon

- Open: The opening price of a stock is the price at which the stock first trades in the market. It is the initial price at which it is traded for a specific time frame. In the stock chart, it is symbolised by the letter "O."
- High: This is the stock's highest price during the day since trading began. In the stock chart, it is denoted by the letter "H."
- Low: This is the stock's lowest price during the day since trading began. In the stock chart, it is denoted by the letter "L."
- Close: The closing price of a stock is the final price at which it traded during the trading session. It is the price at which a stock is closed for a specific time period. In the stock chart, it is denoted by the letter "C."
- Volume: The quantity of an asset or security that changes hands over a specific period of time, usually a day, is referred to as volume. Stock trading volume, for example, refers to the number of shares of a securities traded between its daily open and close.
- Adjusted Close: An adjusted closing price is a stock's closing price that has been adjusted to reflect its worth after accounting for any corporate activities. It is frequently employed for analysing historical returns or performing a comprehensive examination of prior performance.

B. Simple Moving Average

With time series data, a moving average is widely used to smooth out short-term swings and emphasise longer-term trends or cycles. The threshold between short-term and long-term is determined by the application, and the moving average

parameters are adjusted accordingly. It is frequently employed in technical analysis of financial data, such as stock prices, returns, or trade volumes. It is also used in economics to study GDP, employment, and other macroeconomic time series. A moving average is a sort of convolution in mathematics, and as such, it may be considered as an example of a low-pass filter used in signal processing. When applied to non-time series data, a moving average selects higher frequency components without regard for time, however some sort of ordering is usually inferred. When viewed simplistically, it may be thought of simply smoothing the data.

A simple moving average (SMA) is the unweighted mean of the past k data points in financial applications. In science and engineering, however, the mean is generally calculated from an equal number of data points on either side of a centre value. This guarantees that mean variations are matched with data variations rather than being moved in time. The mean over the past k entries of a data-set with n entries is an example of a basic evenly weighted running mean. Let's call those data points $p, p, ..., p$. This might be a stock's closing 12n pricing. SMA_k is the mean of the latest k data points (days in this case) and is computed as:

$$SMA_k = \frac{p_{n-k+1} + p_{n-k+2} + ... + p_n}{k}$$

When calculating the next mean SMAk,next with the same sampling width k the range from $nk+2$ to $n+1$ is considered. A new value $p_n + 1$ comes into the sum and the oldest value $p_n k + 1$ drops out. This simplifies the calculations by reusing the previous mean SMA_k, prev.

$$SMA_{k,next} = \frac{1}{k} \sum_{i=n-k+2}^{n+1} p_i = SMA_{k,prev} + \frac{1}{k}(p_{n+1} - p_{n-k+1})$$

This means that the moving average filter can be computed quite cheaply on real time data with a FIFO / circular buffer and only 3 arithmetic steps.

III. The problem

The data set provided explains the stock prices of ICICI, HDFC, SBI, Infosys, Cognizant, and HCL, as well as the USD-INR exchange rate. In this part, we give the findings from a detailed examination of stock market data.

A. Analysis of individual stock prices

We plot the starting, high, low, and closing values of all stocks using visualisation tools. The following conclusions may be drawn from Figure-3:

- Due to the COVID-19 epidemic, all stocks dropped in March-April 2020.
- Cognizant stock witnessed two large drops, but it regained momentum when the lockdown was lifted.
- HDFC stocks plummeted the most, but showed bullish behaviour and surged back up in the previous fiscal year.
- Following the epidemic, all equities provided consistent returns with the exception of Cognizant, whose price stayed nearly constant.

One the other hand, Figure-1 depicts the USD-INR exchange rate. It can be clearly seen that the dollar rose up when the world was hit by the COVID-19. After hitting the peak around seventy eight rupees dollar has been on a downward trend and currently is around the seventy five rupees mark.

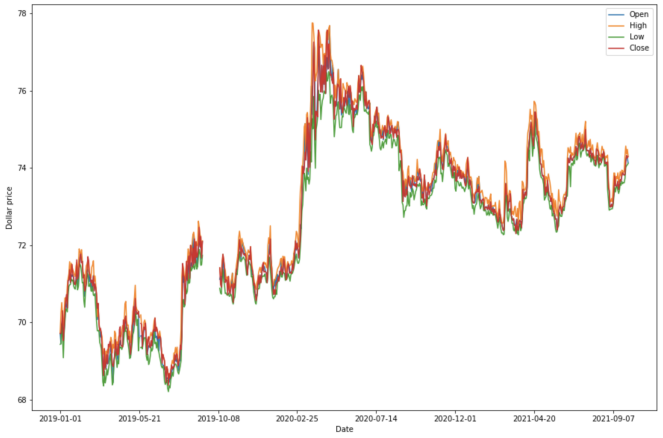


Figure-1: USD-INR exchange rate

B. Sector-wise stock price analysis

From Figure-2 it can be clearly observed that the banking sector stocks have seen a similar trend. As expected, a lot of panic selling happened during the COVID-19 outbreak in India which rightly so caused the dip in price of stocks. Since then, all the stocks in the banking sector, especially HDFC have shown steady growth.

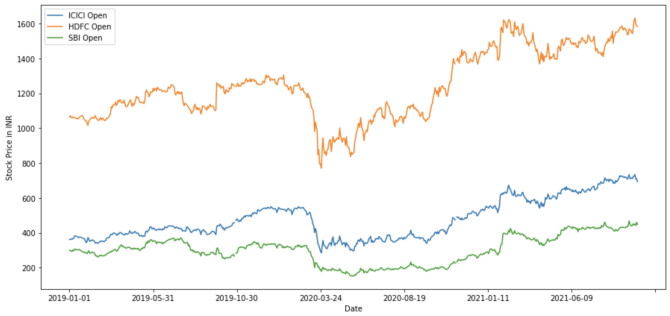


Figure-2: Opening price of banks- ICICI, HDFC and SBI

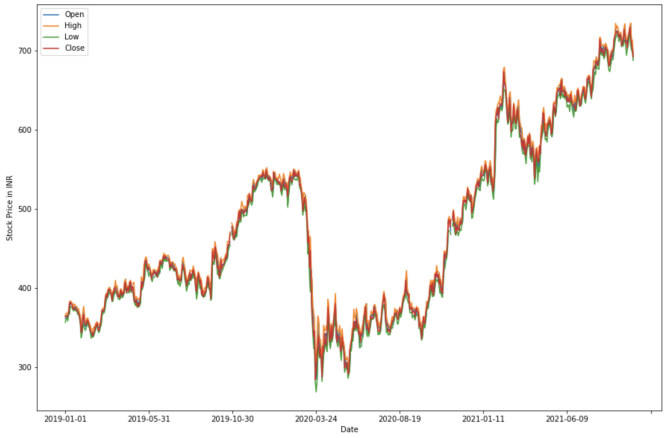


Figure-3 a): Open-High-Low-Close chart of company stocks for ICICI

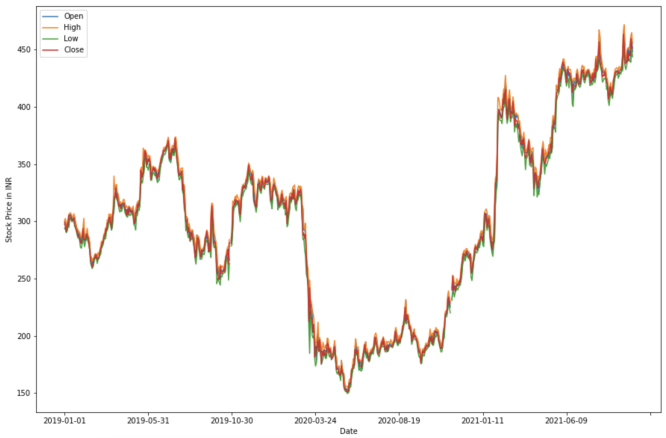


Figure-3 b): Open-High-Low-Close chart of company stocks for SBI

Figure-3 c): Open-High-Low-Close chart of company stocks for HDFC

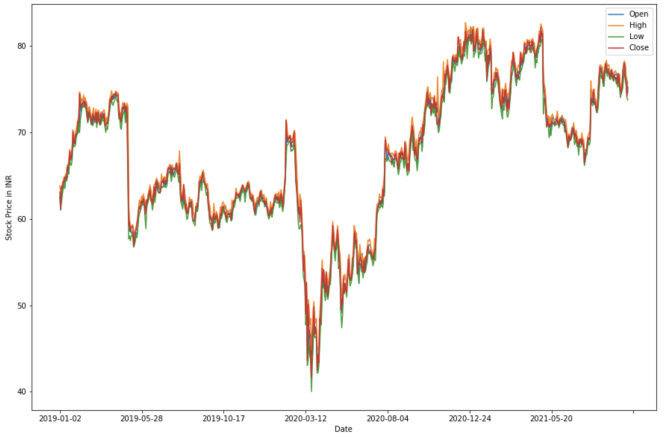


Figure-3 d): Open-High-Low-Close chart of company stocks for Cognizant

Figure-3 e): Open-High-Low-Close chart of company stocks for HCL

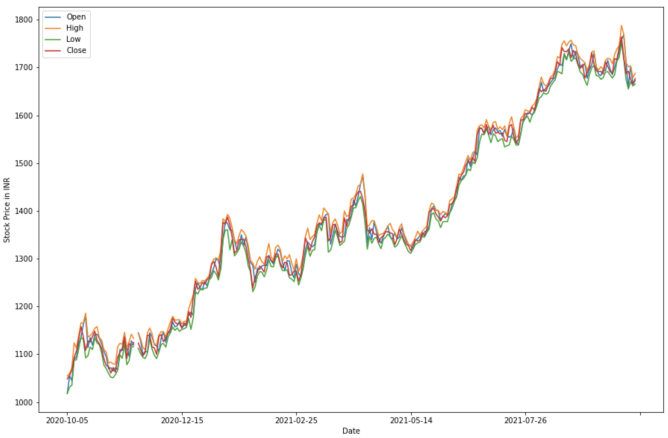


Figure-3 f): Open-High-Low-Close chart of company stocks for Infosys

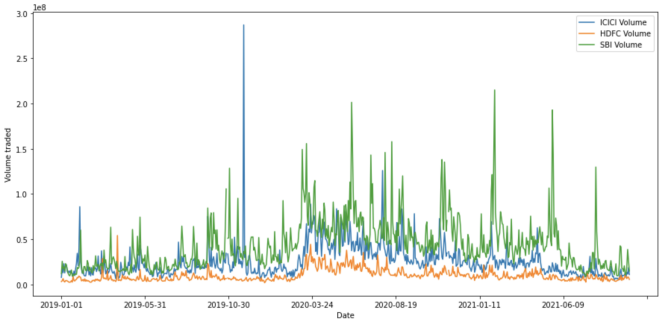


Figure-4: Volume of stocks traded- ICICI, HDFC and SBI

Figure 4 depicts the volume of equities traded in the banking industry. The figure clearly shows that there are several situations where the volume transacted is extremely high. One of the reasons for this is the quarterly reports issued by banks. These reports publish information on how the business is functioning and what its future goals are, which is important information for investors since it helps them decide whether or not to invest in the stock. We can also see that ICICI's traded stock volume increased significantly in November 2019, owing to the announcement of a cut in corporate tax rates.

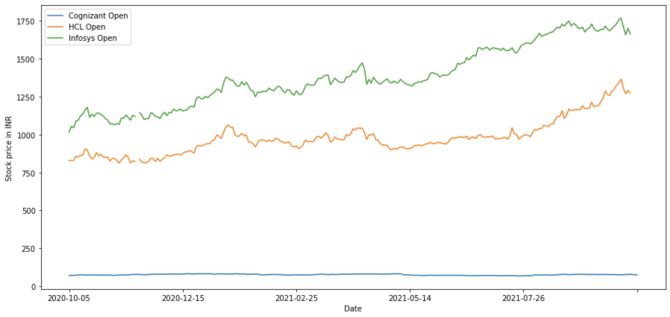


Figure-5: Opening price of banks- Cognizant, HCL and Infosys

From Figure-5 it can be clearly observed that the IT sector stocks have seen a similar trend. Since the data given is after the upliftment of lockdown, one can clearly observe steady growth in the stock prices. But something seems unusual in the stock price of Cognizant. From Figure-3(d) we can conclude that the stock hasn't shown much growth when compared to its competitors.

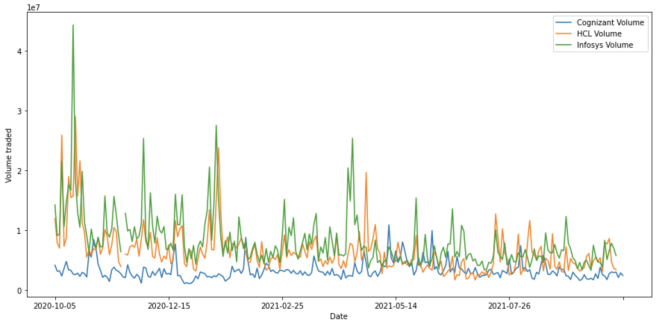


Figure-6: Volume of stocks traded- Cognizant, HCL and Infosys

Figure 6 depicts the volume of equities traded in the IT sector. The figure clearly shows that there are several situations where the volume transacted is extremely high. One of the reasons for this is the quarterly reports issued by banks. These reports publish information on how the business is functioning and what its future goals are, which is important information for investors since it helps them decide whether or not to invest in the stock. We can also see that the traded stock volume of Infosys soared in October 2020, owing to the announcement of a successful second quarter. Even while it may appear strange that the stock price dropped by a tiny amount after announcing strong results, this is common when a large number of investors sell the company, anticipating that the stock has peaked. Throughout the year we can observe that the volume of traded Cognizant stock was low because maybe most of the investors are waiting for the stock to shoot up so that they can sell it at a higher price, than what they bought at.

C. *Total money traded trend*

By multiplying the closing price by the total volume traded, the essence of total traded value may be represented. Some spikes can be seen in the banking plots. Economic data, geopolitical events, and market mood are just a few of the reasons that may cause the stock market to move drastically in one direction or the other. For example, the ICICI bank stock had the greatest increase towards the end of 2019 as a result of the government's announcement of a cut in corporate tax rates.

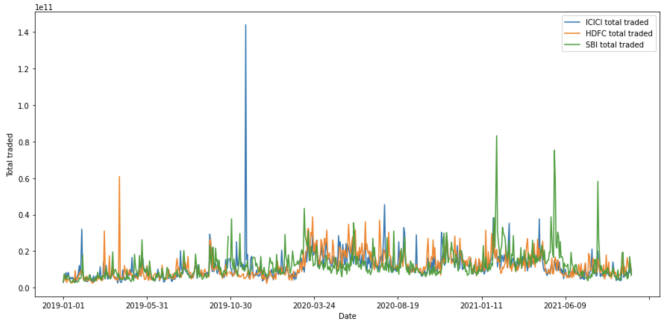


Figure-7: Total money traded trend- Cognizant, HCL and Infosys

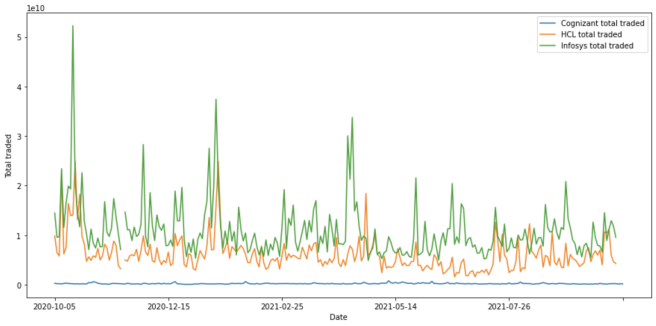


Figure-8: Total money traded trend- Cognizant, HCL and Infosys

D. Sector-wise Scatter Matrix

In our database, we have three banks and three IT enterprises. We assume that the increase and decrease in stock prices of firms in the same industry will be connected. To demonstrate this, we may create scatter plots for each sectors.

From Figure-9 we can clearly see high correlation amongst the banking sector bonds. It can be understood as the change in government policies and ups and downs in economy affect all the banks. Same goes for the information technology sector.

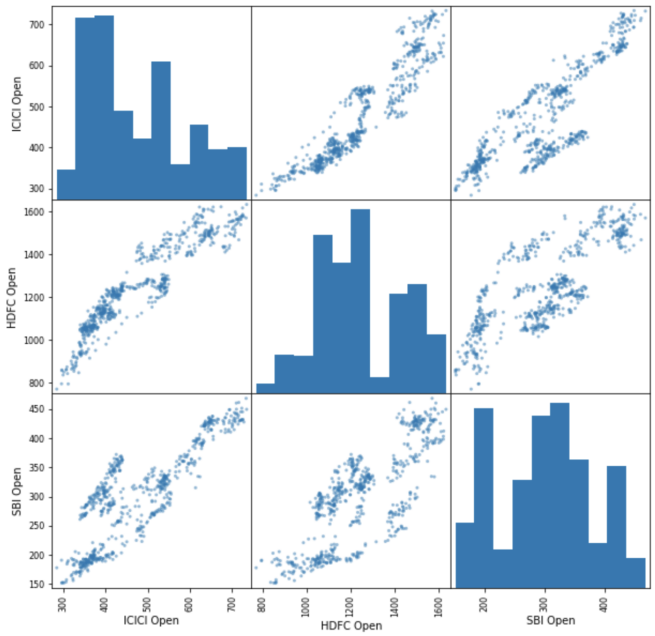


Figure-9: Scatter Matrix- ICICI, HDFC and SBI

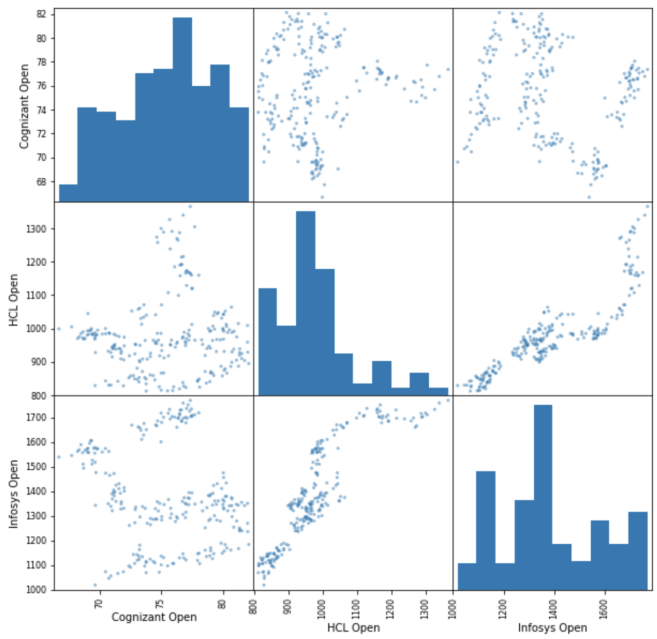


Figure-10: Scatter Matrix- Cognizant, HCL and Infosys

E. Sector-wise daily returns

The returns on a stock may be computed as the change in price of the stock over time, which can be expressed as a percentage change or as a price change. We can calculate the returns of the supplied stocks by computing the percentage change in the stock's closing value. The returns are plotted in the lineplot below.

The line plot makes it difficult to assess and quantify the returns. To have a better understanding, we may assess the volatility of the returns. Volatility reflects the extent to which price moves. A highly volatile stock has a price that varies wildly—hits new highs and lows or moves unpredictably. Low volatility is associated with a stock that maintains a reasonably constant price.

We can calculate volatility by plotting the distribution of returns. SBI is less volatile than HDFC and ICICI in the banking sector, and it provides consistent returns. HCL is less volatile than Infosys equities in the IT industry.

Box charts are another approach to see the results. Box plots show variance in statistical population samples without making any assumptions about the underlying statistical distribution. According to the box plots, ICICI bank has the greatest mean return, followed by SBI and HDFC, respectively. According to the IT sector box plot, the mean return of HCL and Infosys is nearly identical, although HCL returns are more spread out.

F. Cumulative returns

A cumulative return on an investment is the total amount gained or lost over time, regardless of the length of time involved. It is the overall change in the price of an investment over a certain time period. We can plot the cumulative returns for both banking and IT sector.

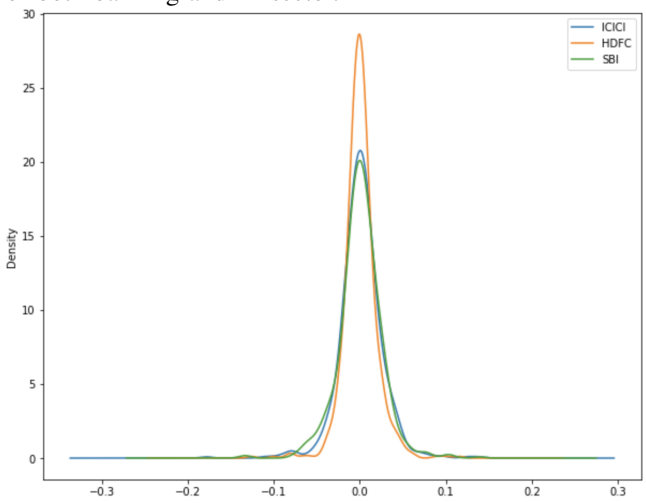


Figure-11: Returns plot of Banking sector stocks

IV. Conclusions

Only roughly 2,000 pulsars have been detected so far. For their celestial research, scientists and astronomers benefit from the job of establishing whether a star is a pulsar or not. The provided data was used to describe the link between

various qualities and determine which ones were the most advantageous for this issue statement. Using the EDA and Support Vector classifier approach, the link between predictors and regressors may be better understood. It aids in the explanation of the underlying pattern and the prediction of accurate outcomes. In the future, GridSearchCV and RandomSearchCV can be utilised to find the best imputation approach and do hyper-parameter selection.

REFERENCES

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

[2] Kevin P. Murphy, "Machine Learning, A Probabilistic Perspective"

[3] Yunus Koloğlu, Hasan Birinci, Sevde Ilgaz Kanalmaz, Burhan Özyılmaz, "A Multiple Linear Regression Approach For Estimating the Market Value of Football Players in Forward Position"

[4] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning. New York :Springer, 2001, pp. 417-434