NAME: PRIYANKA PRAKASH BABU

UTD ID: PXP190044

COURSE: MIS 6357 – ADVANCED BUSINESS ANALYTICS WITH R

PROJECT TITLE: FAKE JOB POSTINGS PREDICTION

PROFESSOR: SOURAV CHATTERJEE

THE UNIVERSITY OF TEXAS AT DALLAS

NAVEEN JINDAL SCHOOL OF MANAGEMENT

SUMMER 2020

Contents

1. Objective 2
2. Introduction 2
3. Problem Definition 2
4. Exploratory Data Analysis 2
4.1 Dataset Information 2
4.2 Different Plots to understand classification of dataset based on different parameters 3
5. Feature Extraction from Job Description 8
5.1 Text Pre-Processing 8
5.2 Feature Extraction 9
6. Splitting the Dataset: 9
6.1 Training Partition: 9
6.2 Validation Partition: 9
7. Comparing Multiple Algorithms 9
7.1. Before Text Analytics 9
7.1.1 Logistic Regression 10
7.1.2. Random Forest 11
7.1.3.Naïve Bayes 13
7.2. After Text Analytics 14
7.2.1 Support Vector Machines (SVM) 14
7.2.2. Random Forest 16
7.2.3. Stochastic Gradient Boosting 17
8. Conclusion 21
9.Appendix 22
10. References : 33

1. Objective

The objective of the dataset is to predict whether a job posting is fake or not, based on certain features included in the dataset. Increasing the accuracy in detecting the prediction of a fake job by building a classification model using the text features in the job description column only to predict the fake job postings is the main goal of this project.

2. Introduction

Today, we have many websites which post millions of jobs positions. Of which many are fake. It is important to be able to differentiate legitimate job opportunities from fake opportunities that could result in one being scammed if decided to pursue them. Our goal for this project is to solve this business problem -to find job descriptions which are fraudulent using Machine learning algorithms - classification techniques.

The data consists of both textual information and meta-information about the jobs. The dataset can be used to create classification models which can learn the job descriptions which are fraudulent.

The outcome of this report also involves exploratory data analysis to gain insights into the data and identify key traits/features of job description which are fraudulent in nature.

3. Problem Definition

The dataset that is obtained from Kaggle. The objective of this dataset is to predict if a job posting is fraudulent or not. The dataset consists of several predictor variables and one target variable, which is the outcome variable. Predictor variables include job title, job location, job department, salary range, company profile, employment type, required experience, required education, industry, function, and so on. Target variable includes one column "fraudulent". It has a binary value in it(1 or 0). “0” indicates a genuine job. “1” indicates that the job description is fraudulent. This dataset is taken from Kaggle, contains 18K observations and has 18 defining attributes and 1 outcome (as Fraudulent 1 or 0). Of the entire list of 18k records, there are about 800 fraudulent jobs. It was interesting to note that there were couple of duplicate terms in the dataset and had to be altered

I have divided 90% as training dataset and 10% data as validation dataset. I have tried to find algorithms which work best with the available dataset.

4. Exploratory Data Analysis

4.1 Dataset Information

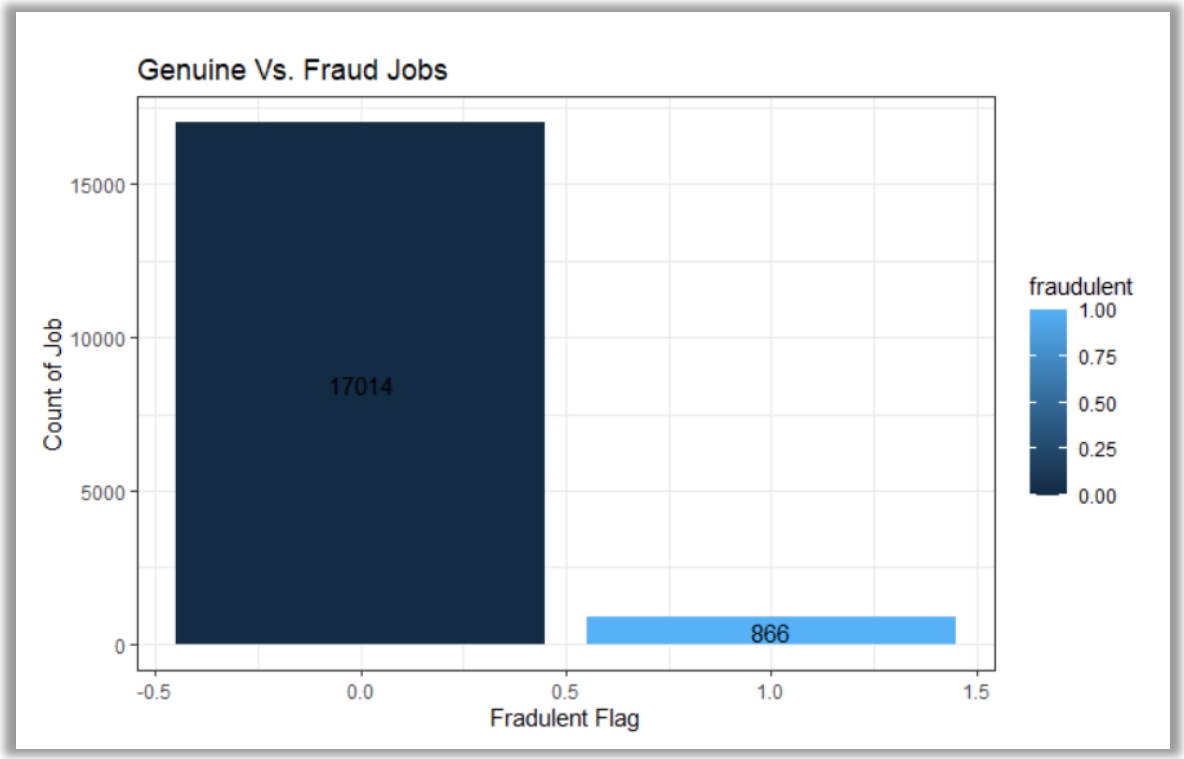
Description for all the variables is given below. 17880 obs. of 18 variables.

Data Dictionary:

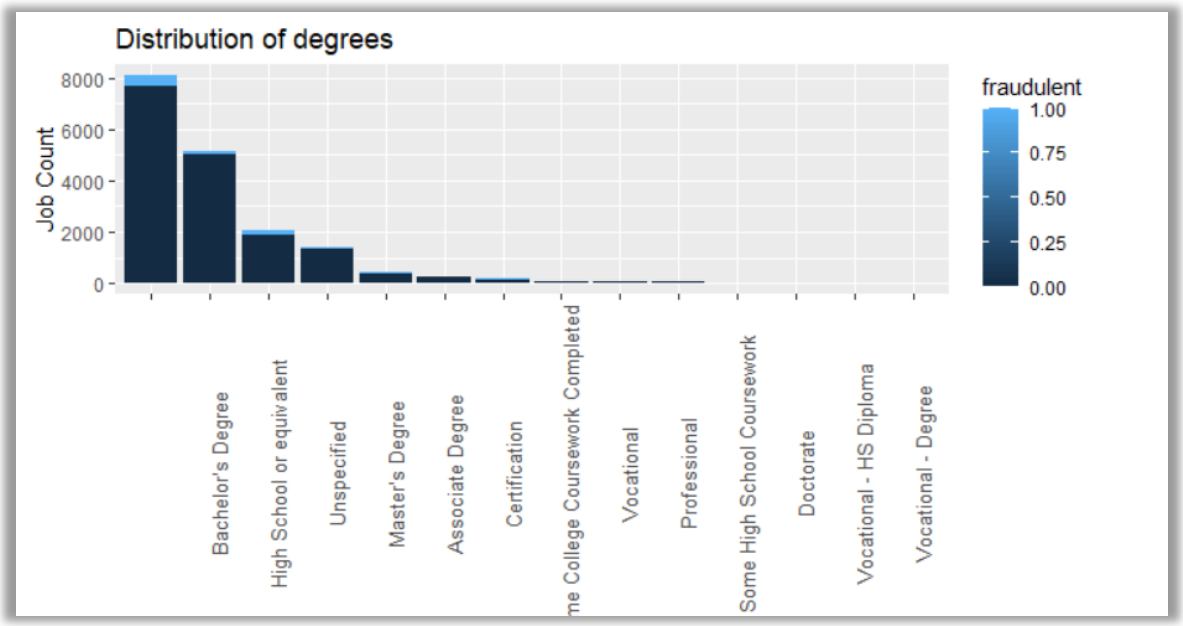
Sl. no.	Attribute Name	Description	Type of Attribute
1	job_id	Unique Job identification number	Integer
2	title	Title of the job	Character
3	location	Geographic location of the job ad	Character
4	department	Corporate Department (ex. Sales)	Character
5	salary_range	Indicative salary range offered by the company	Character
6	company_profile	A brief description about the company	Character
7	telecommuting	True for telecommuting positions.	Integer
8	has_company_logo	True if company logo is present	Integer
9	has_questions	True if screening questions are present	Integer
10	employment_type	Full-type, Part-time, Contract, etc.	Character
11	required_experience	Executive, Entry level, Intern, etc.	Character
12	required_education	Doctorate, Master's Degree, Bachelor, etc.	Character
13	industry	Automotive, IT, Health care, Real estate, etc.	Character
14	function	Consulting, Engineering, Research, Sales etc.	Character
15	fraudulent	target - Classification attribute. (Binary)	Integer

4.2 Different Plots to understand classification of dataset based on different parameters

a. I tried to plot Job count Vs Fraud Flag and found that there is a total of 866 fraud job postings.

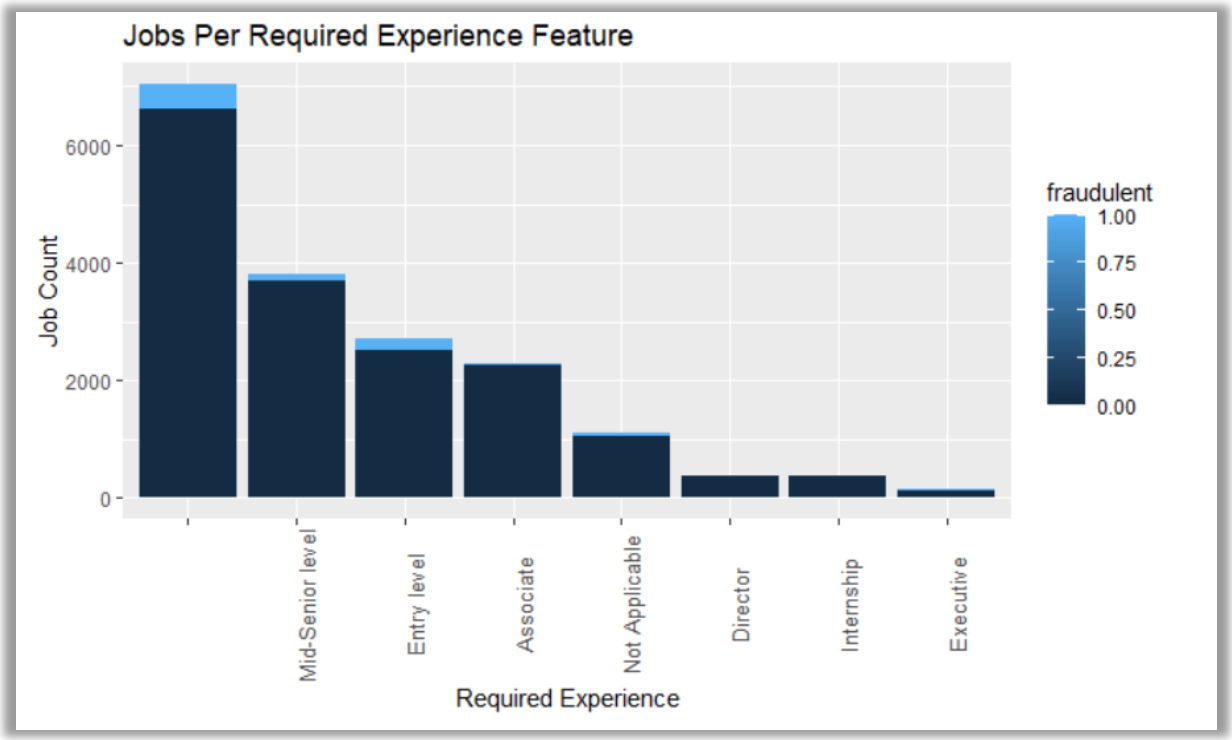


b. Plot- the job count per required education colored based on the job classification.



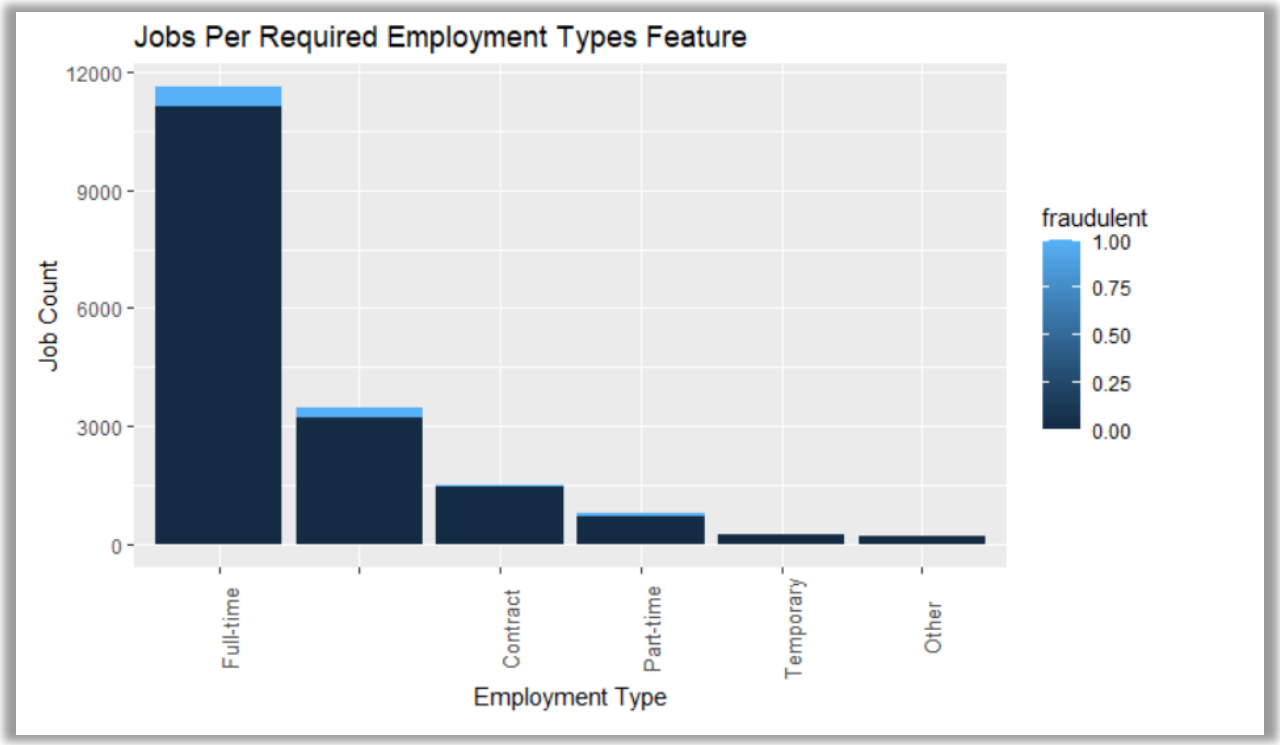
It was interesting to note that the max count column (about 8000 jobs) did not have education listed at all. The next best job count is for bachelor's degree. Few of the required education categories does not have any fake job postings like professional, doctorate etc.

c. To further investigate the data, the plot with jobs per required education colored by required experience was used.



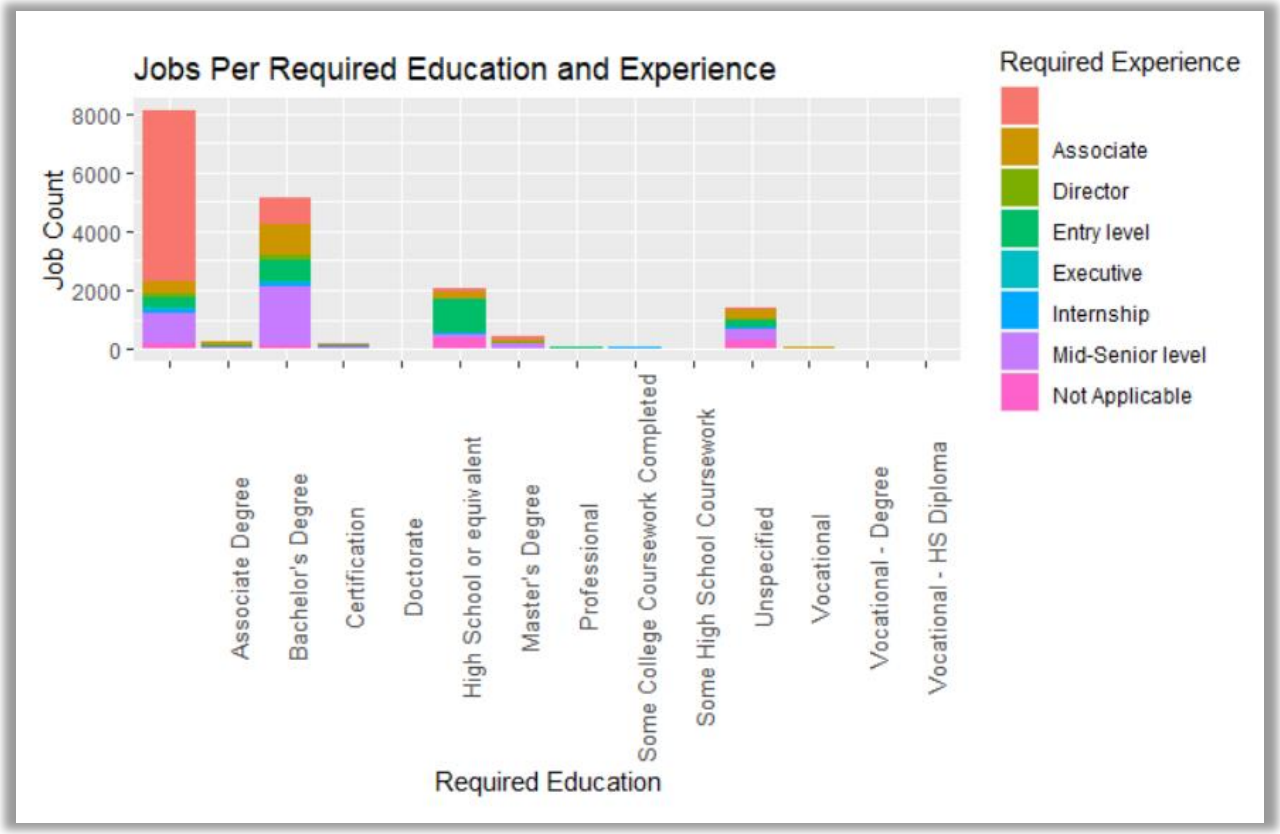
About 7000 jobs does not have any experience listed in them. The next best job count require mid-senior level job experience. All types in this feature has fake jobs listed in them.

d. Plot below shows Job count Vs Employment Types



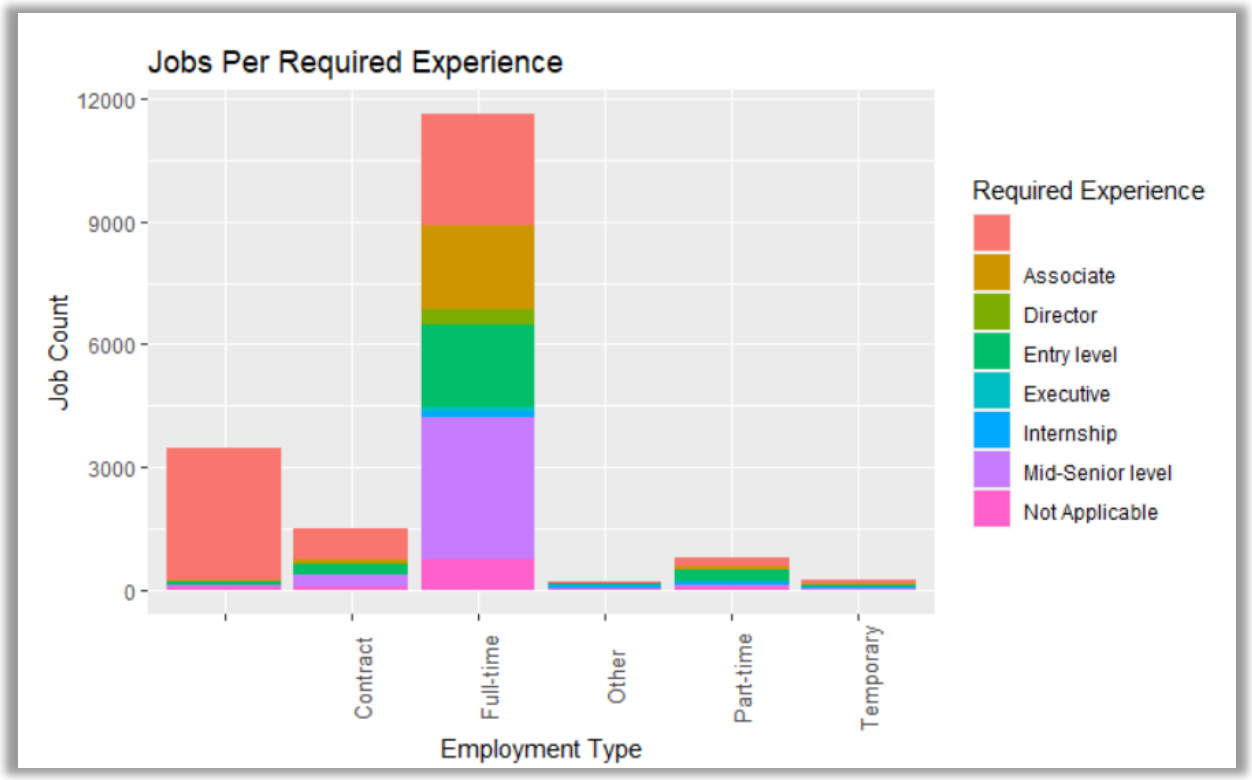
Majority of the jobs are full-time, the next best is not listed. Only temporary job type does not have any fake listings.

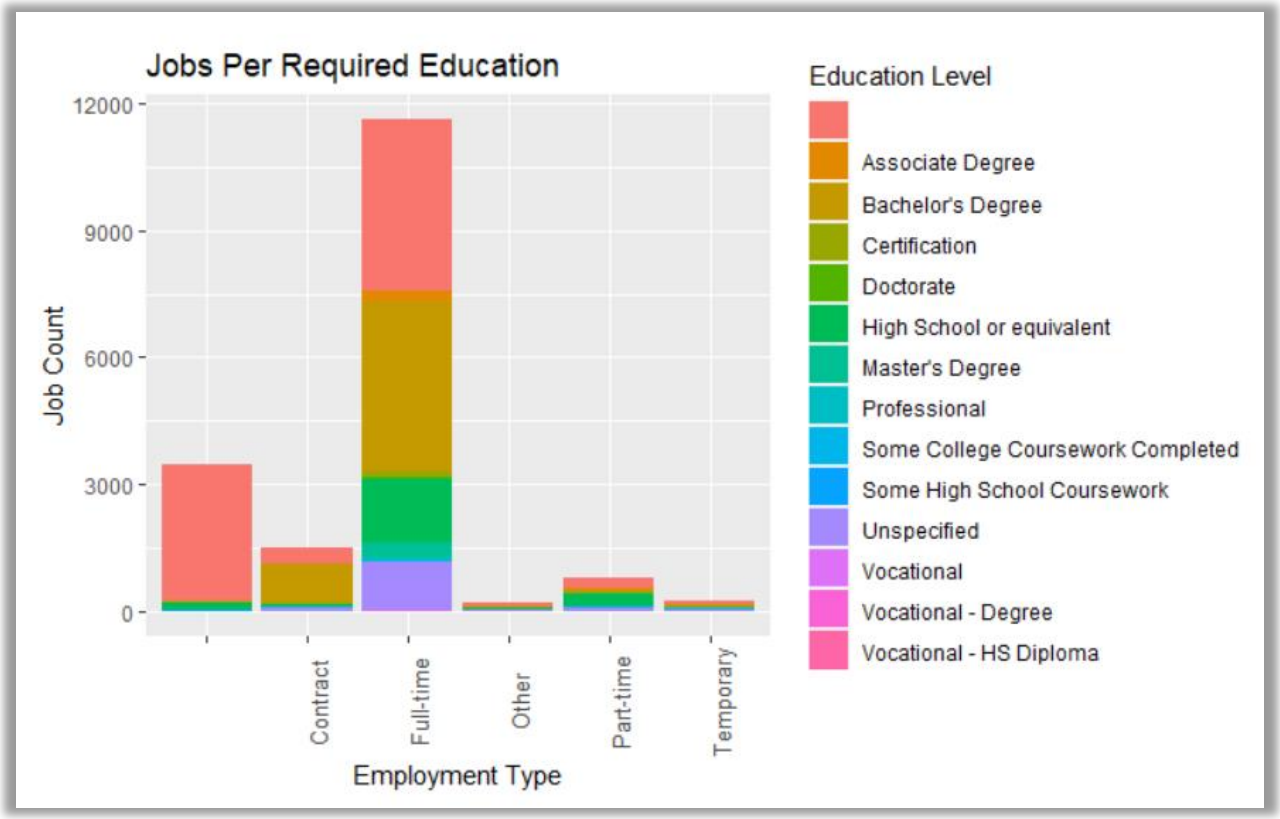
e. Below is a plot of Jobs Vs Education and Experience



Majority of the job postings that had unlisted required education feature had unlisted required experience feature as well. Similarly, majority of the bachelor's degree jobs required mid-senior job experience. I think this required experience column should have been the experience level for the job posting rather than required experience. Most of the entry level jobs required high school equivalent which makes sense.

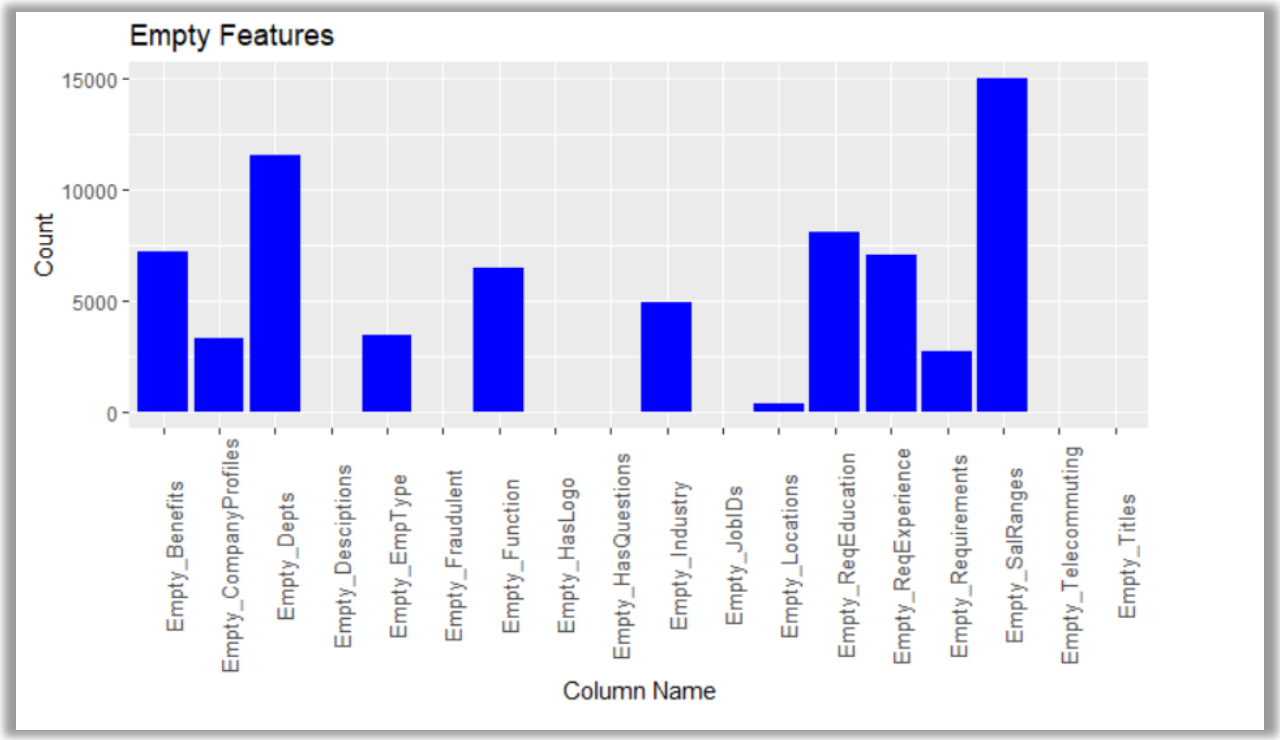
f. Below is a plot of Jobs Vs Experience and Employment type


Jobs Per Required Experience
Job Count
12000
9000
6000
3000
0
Contract
Full-time
Other
Part-time
Temporary
Employment Type
Required Experience
Associate
Director
Entry level
Executive
Internship
Mid-Senior level
Not Applicable


Jobs Per Required Education
Job Count
12000
9000
6000
3000
0
Contract
Full-time
Other
Part-time
Temporary
Employment Type
Education Level
Associate Degree
Bachelor's Degree
Certification
Doctorate
High School or equivalent
Master's Degree
Professional
Some College Coursework Completed
Some High School Coursework
Unspecified
Vocational
Vocational - Degree
Vocational - HS Diploma

It is interesting to note that most of the unlisted job types also have unlisted required education and experience level. All types of experience levels and educational requirements had full time opportunities.

g. A plot was used to better understand what features are empty for different jobs.



There are a total of 7 features which does not have any empty values in them including has questions, job ID's, descriptions, fraudulent flag, has logo, telecommuting, benefits, and titles. There are several thousand jobs postings that have other empty features.

5. Feature Extraction from Job Description

Using Text Pre-Processing and Feature extraction, the key traits in the job description column that are fraudulent in nature is identified.

5.1 Text Pre-Processing

To look for the most frequent words, the most common approaches is to clean the text and tokenize them.

a. Lower Case- The entire description data is normalized by converting all characters to lower case.
b. Alpha-Numeric Characters -These special characters does not have any influence on predicting the fake job description and hence have deleted from the description data.

c. Stop Words - Words which are frequent and provide very little information are called as “stop words”. In the tm()tm() package, there are about 174174 common stop words like "I", "the", "he'll" etc. These words do not have any predictive capability and hence are deleted.
d. White Spaces & Stemming - Sometimes, an extra white space can cause a word to be interpreted incorrectly as the previous transformations can induce such white spaces. Those are deleted. Stemming is a process of identifying different forms of the same word and reducing them. For example, "love loving loved", all three have the same meaning and can be interpreted as one. This is achieved using SnowballC package in R.

5.2 Feature Extraction

The function “TermDocumentMatrix” from the tm() package is used for Feature extraction. From the output of plot, it can be observed that some words are not complete. For example, "manage" is shown as "manag". The reason could be one of the operations performed before feature extraction may have deleted the last character. The size of the word represents its frequency. The larger the word, the higher the frequency. It was found that the top three frequent words are word, will, and team.

6. Splitting the Dataset:

To address the overfitting problem, I have divided the entire dataset into two partitions : a training partition and a validation partition. I have developed the model using only one of the partitions and then used the trained model to make predictions on other partition and compare the predicted value with the pre-assigned label.

6.1 Training Partition:

The largest partition that has data used to build the various models is training partition. We generally use one training partition to develop multiple models.

6.2 Validation Partition:

The validation partition is used to analyze the predictive performance of each model which can be used to compare different models. The validation partition may be used in an automated fashion to tune and improve the model in some algorithms like classification, SVM, k-nearest neighbors.

Splitting the dataset is a very important step for supervised machine learning models.

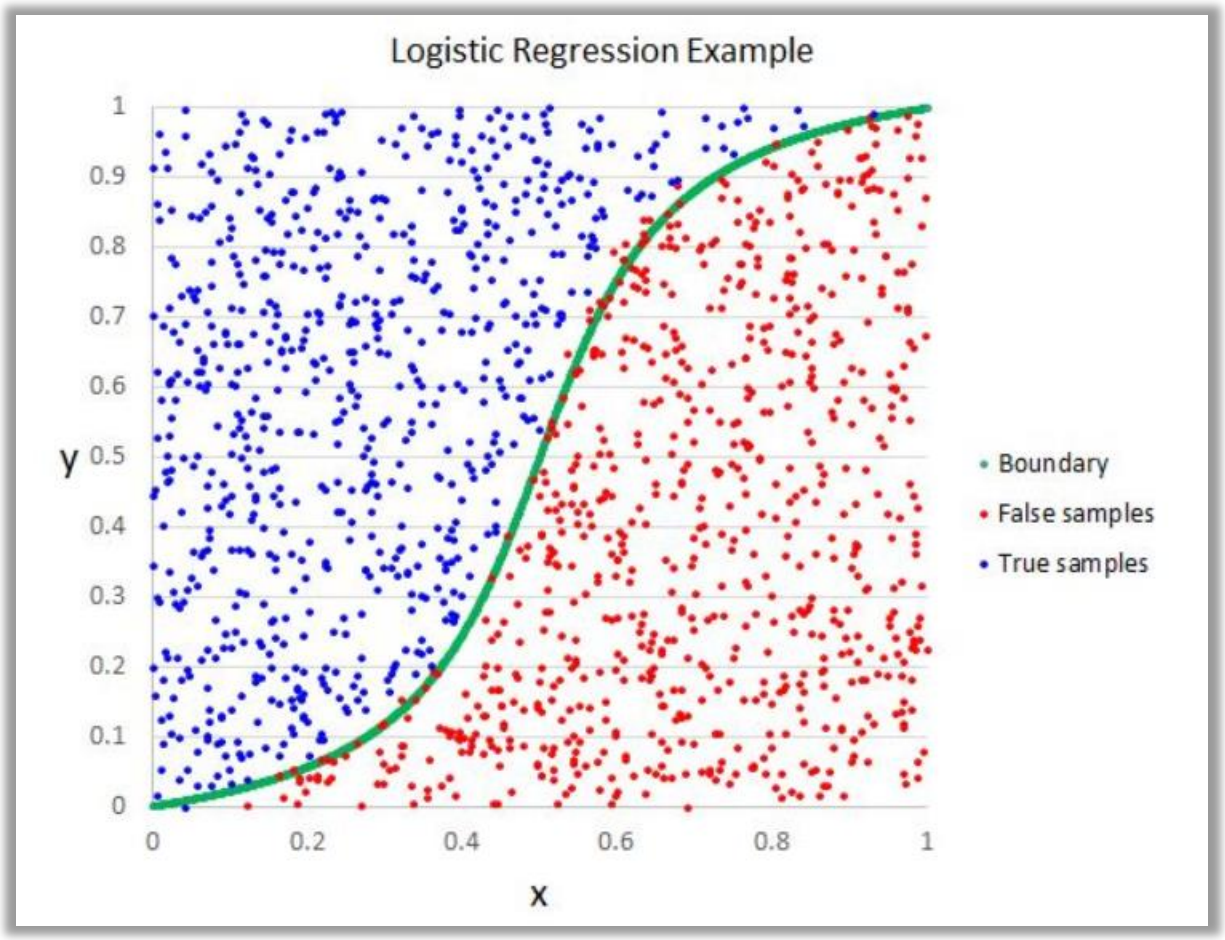
7. Comparing Multiple Algorithms

7.1. Before Text Analytics

Before conducting text analytics, I considered few variables which I found important based on my domain understanding. The variables namely are : telecommuting, has_company_logo, has_questions, employment_type, required_experience, required_education, industry and function.

Using the above variables, I conducted the below algorithms:

7.1.1 Logistic Regression



Logistic regression, also called a logit model, is used to model dichotomous outcome variables. In the logit model the log odds of the outcome is modeled as a linear combination of the predictor variables.

a. I have adjusted the threshold probability to accurately classify the fake jobs.
b. Below is the output of the model when threshold is 0.02:

	FALSE	TRUE
0	1982	1420
1	21	152

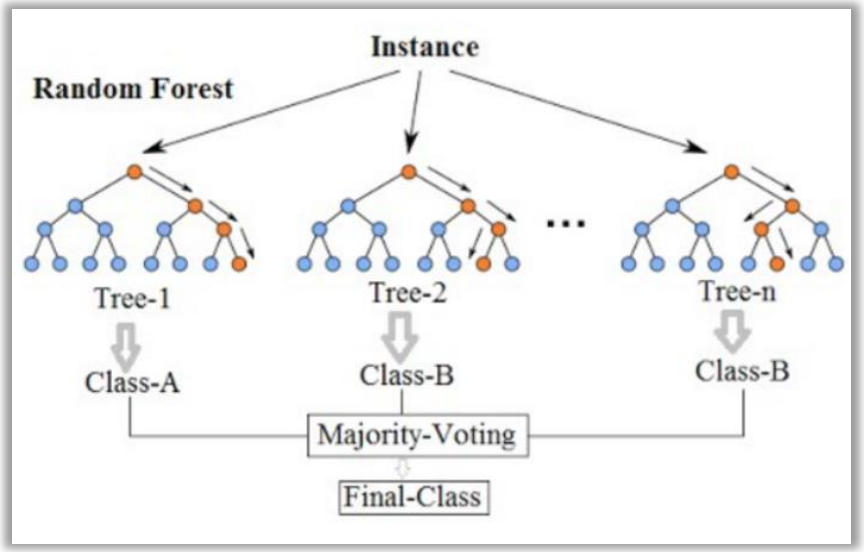
We can infer that out of 173 fake jobs we are able to classify 152 accurately. We are misclassifying 1982 genuine jobs as fake jobs. Accuracy is 43.972 %((1420+152)/3575)

c. Below is the output when threshold probability to 0.01

	FALSE	TRUE
0	796	2606
1	8	165

On adjusting the, we can see that the sensitivity of identifying the fake job is increased. We can now see that 2606 jobs are now classified accurately as genuine. Accuracy is 77.75 % ((2606+165)/3575)

7.1.2. Random Forest



Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. Basically, random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems.

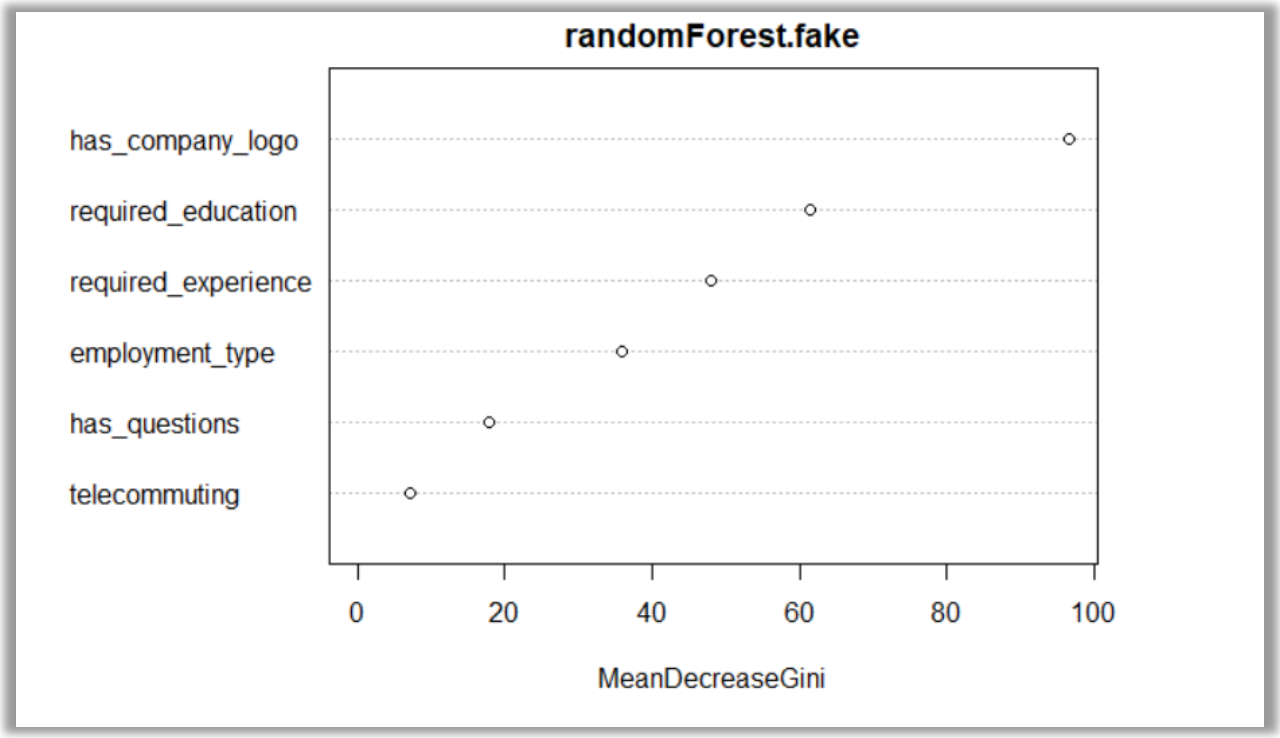
Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node.

Below is the analysis of output :

a. Sensitivity of identifying genuine jobs is 95.506.
b. Accuracy is 95.46 %
c. The No Information Rate is 0.9958, even though this value is very low, we can notice that out of 3402 genuine jobs, the model is predicting 3400 genuine jobs accurately . It is important that we do not wrongly classify a genuine job as fake.
d. Mean Decrease in Gini is the average (mean) of a variable's total decrease in node impurity, weighted by the proportion of samples reaching that node in each individual decision tree in the random forest. This is effectively a measure of how important a variable is for estimating the value of the target variable across all the trees that make up the forest. A higher Mean Decrease in Gini indicates higher variable importance. It can be noticed that the variables – "has_company_logo" is playing a significant role in deciding whether the job is fake or not since it has a higher value . Also the variable "telecommuting" is having lowest meandecreasegini hence has least significance in deciding whether the job is fake or not

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3400    2
         1  160   13

               Accuracy : 0.9547
                 95% CI : (0.9473, 0.9613)
    No Information Rate : 0.9958
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1316

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.95506
            Specificity : 0.86667
         Pos Pred Value : 0.99941
         Neg Pred Value : 0.07514
             Prevalence : 0.99580
         Detection Rate : 0.95105
   Detection Prevalence : 0.95161
      Balanced Accuracy : 0.91086

       'Positive' Class : 0
```

	MeanDecreaseGini
telecommuting	7.050605
has_company_logo	96.667776
has_questions	17.791426
employment_type	35.954461
required_experience	48.029361
required_education	61.549101



7.1.3.Naïve Bayes

Naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models. But they could be coupled with Kernel density estimation and achieve higher accuracy levels.

Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by

evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

a. From the below output, we can notice that the sensitivity has now increased to 99.70% identifying genuine jobs accurately.
b. Accuracy is 94.99 %

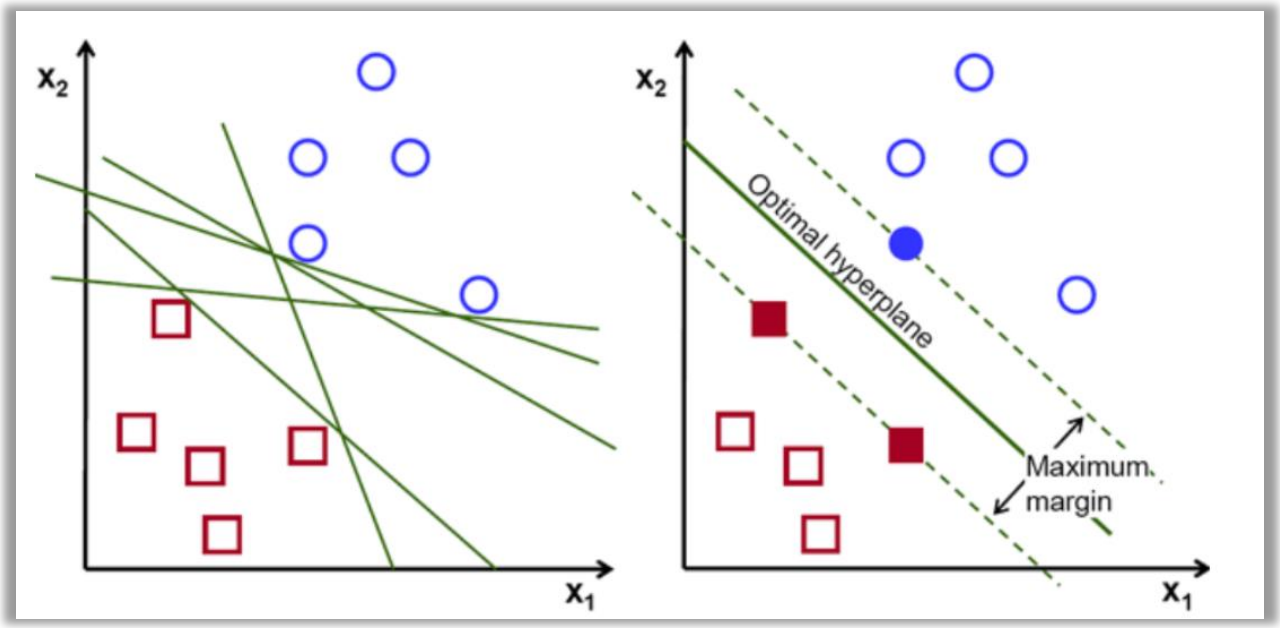
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3392  169
         1   10    4

               Accuracy : 0.9499
                 95% CI : (0.9423, 0.9568)
    No Information Rate : 0.9516
    P-Value [Acc > NIR] : 0.6968

                  Kappa : 0.0358

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.99706
            Specificity : 0.02312
         Pos Pred Value : 0.95254
         Neg Pred Value : 0.28571
             Prevalence : 0.95161
         Detection Rate : 0.94881
   Detection Prevalence : 0.99608
      Balanced Accuracy : 0.51009

       'Positive' Class : 0
```

7.2. After Text Analytics

To know which algorithm will work better with our dataset, we will have to compare a few and choose the one with “best score”

Models:

7.2.1 Support Vector Machines (SVM)



Support Vector Machine is a classification and regression algorithm. The objective of SVM is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes.

a. A subset of the dataset has been used to test run to get an idea of timing instead of waiting for hours to get the simulations to run to completion.
b. The SVM model is tuned for the cost (C) of the linear classifier. This is specified in the tuneGrid parameter and the cost is varied from 0.01 to 20 .
c. 5 – fold cross validation is chosen to run SVM
d. The model had the best accuracy at Cost C = 0.01
e. Accuracy for this model was found to be 96%
f. Below is the output of SVM:

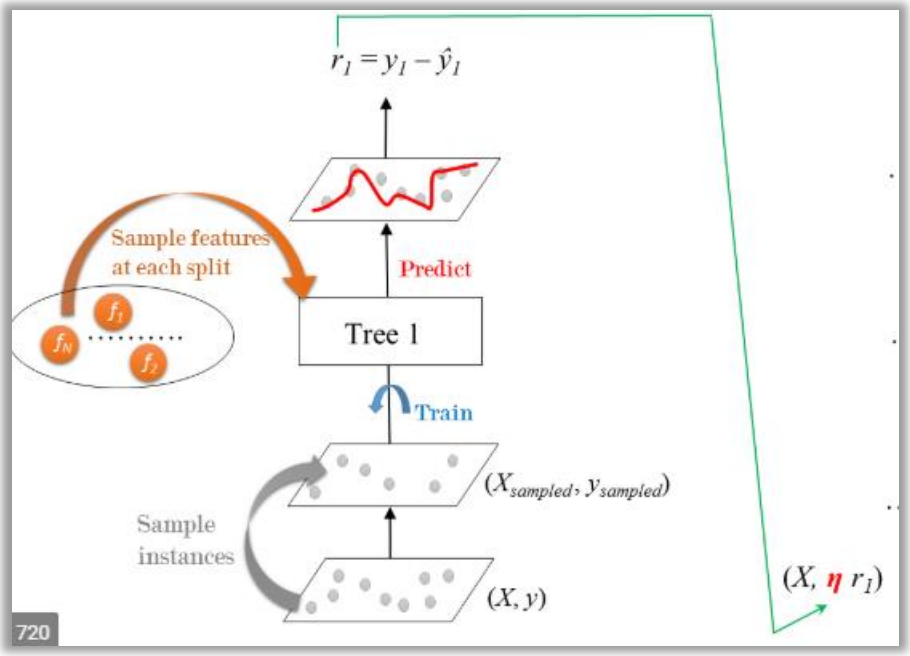
```
+ Fold1: C=0.01
- Fold1: C=0.01
+ Fold2: C=0.01
- Fold2: C=0.01
+ Fold3: C=0.01
- Fold3: C=0.01
+ Fold4: C=0.01
- Fold4: C=0.01
+ Fold5: C=0.01
- Fold5: C=0.01
Aggregating results
Fitting final model on full training set
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1675   31
         1   35   47

               Accuracy : 0.9631
                 95% CI : (0.9533, 0.9713)
    No Information Rate : 0.9564
    P-Value [Acc > NIR] : 0.08904

                  Kappa : 0.5682

 Mcnemar's Test P-Value : 0.71192

            Sensitivity : 0.9795
            Specificity : 0.6026
         Pos Pred Value : 0.9818
         Neg Pred Value : 0.5732
             Prevalence : 0.9564
         Detection Rate : 0.9368
   Detection Prevalence : 0.9541
      Balanced Accuracy : 0.7910

       'Positive' Class : 0
```

7.2.2. Random Forest

a. For this problem, 5 -fold cross validation was chosen.
b. The tuning parameter " mtry" (number of randomly chosen variables from predictors list) was varied from 1 to 100 and the tuning was done.
c. This random forest tuning process took approximately 10 hours to complete.
d. The best tune parameter was mtry = 100 , and this was used to predict the results on the validation dataset.
e. Accuracy for this model was found to be 97.9%
f. Below is the output of Random Forest:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1710   37
         1    0   41

Accuracy : 0.979
95% CI : (0.972, 0.985)
No Information Rate : 0.956
P-Value [Acc > NIR] : 1.12e-07

Kappa : 0.679

Mcnemar's Test P-Value : 3.25e-09

            Sensitivity : 1.000
            Specificity : 0.526
         Pos Pred Value : 0.979
         Neg Pred Value : 1.000
             Prevalence : 0.956
         Detection Rate : 0.956
   Detection Prevalence : 0.977
      Balanced Accuracy : 0.763

       'Positive' Class : 0
```

7.2.3. Stochastic Gradient Boosting



Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The difference between GBM and random forests is the way the trees are built, their order, and the way the results are combined. Some of the advantages of GBM includes flexibility, handling missing data, predictive accuracy, and no data pre-processing required. On the other hand, GBM algorithms required larger grid search tuning compared to the random forests.

a. For this problem, the values for gbm tuning parameters were chosen to be similar as in random forests analysis (for example, number of trees, interaction depth etc.) wherever applicable in order to have a fair comparison between them.
b. First, a subset of the training data was used to train the model to get the estimate of the time and later full dataset was used to train the model.
c. The best tune set was at iteration depth = 25
d. Accuracy for this model was found to be 97.65%
e. Below is the output of Stochastic Gradient Boosting :

Iter	TrainDeviance	ValidDeviance	StepSize	Improve
1	0.3448	-nan	0.1000	0.0204
2	0.3239	-nan	0.1000	0.0079
3	0.3095	-nan	0.1000	0.0060
4	0.2980	-nan	0.1000	0.0045
5	0.2883	-nan	0.1000	0.0033
6	0.2795	-nan	0.1000	0.0032
7	0.2719	-nan	0.1000	0.0025
8	0.2656	-nan	0.1000	0.0019
9	0.2590	-nan	0.1000	0.0017
10	0.2536	-nan	0.1000	0.0013
20	0.2067	-nan	0.1000	0.0012
40	0.1547	-nan	0.1000	0.0001
60	0.1200	-nan	0.1000	0.0001
80	0.0966	-nan	0.1000	-0.0001
100	0.0800	-nan	0.1000	-0.0002
120	0.0670	-nan	0.1000	-0.0000
140	0.0576	-nan	0.1000	-0.0001
150	0.0536	-nan	0.1000	-0.0000

Iter	TrainDeviance	ValidDeviance	StepSize	Improve
1	0.3464	-nan	0.1000	0.0207
2	0.3267	-nan	0.1000	0.0091
3	0.3123	-nan	0.1000	0.0058
4	0.3018	-nan	0.1000	0.0040
5	0.2912	-nan	0.1000	0.0038
6	0.2826	-nan	0.1000	0.0029
7	0.2735	-nan	0.1000	0.0033
8	0.2667	-nan	0.1000	0.0018
9	0.2603	-nan	0.1000	0.0021
10	0.2545	-nan	0.1000	0.0017
20	0.2078	-nan	0.1000	0.0010
40	0.1544	-nan	0.1000	0.0003
60	0.1204	-nan	0.1000	-0.0001
80	0.0967	-nan	0.1000	-0.0002
100	0.0790	-nan	0.1000	-0.0001
120	0.0659	-nan	0.1000	-0.0002
140	0.0562	-nan	0.1000	-0.0001
150	0.0523	-nan	0.1000	-0.0001

Iter	TrainDeviance	ValidDeviance	StepSize	Improve
1	0.3478	-nan	0.1000	0.0178
2	0.3278	-nan	0.1000	0.0076
3	0.3111	-nan	0.1000	0.0064
4	0.3003	-nan	0.1000	0.0038
5	0.2894	-nan	0.1000	0.0038
6	0.2815	-nan	0.1000	0.0029
7	0.2744	-nan	0.1000	0.0023
8	0.2664	-nan	0.1000	0.0025
9	0.2598	-nan	0.1000	0.0022
10	0.2543	-nan	0.1000	0.0017
20	0.2057	-nan	0.1000	0.0011
40	0.1506	-nan	0.1000	0.0002
60	0.1179	-nan	0.1000	0.0001
80	0.0951	-nan	0.1000	-0.0002
100	0.0787	-nan	0.1000	-0.0001
120	0.0652	-nan	0.1000	-0.0001
140	0.0551	-nan	0.1000	-0.0000
150	0.0506	-nan	0.1000	-0.0001

Iter	TrainDeviance	ValidDeviance	StepSize	Improve
1	0.3419	-nan	0.1000	0.0202
2	0.3251	-nan	0.1000	0.0077
3	0.3107	-nan	0.1000	0.0058
4	0.2998	-nan	0.1000	0.0038
5	0.2890	-nan	0.1000	0.0041
6	0.2806	-nan	0.1000	0.0024
7	0.2716	-nan	0.1000	0.0032
8	0.2651	-nan	0.1000	0.0021
9	0.2570	-nan	0.1000	0.0029
10	0.2513	-nan	0.1000	0.0015
20	0.2040	-nan	0.1000	0.0009
40	0.1497	-nan	0.1000	0.0002
60	0.1159	-nan	0.1000	-0.0000
80	0.0941	-nan	0.1000	-0.0001
100	0.0781	-nan	0.1000	-0.0002
120	0.0655	-nan	0.1000	0.0000
140	0.0555	-nan	0.1000	-0.0001
150	0.0512	-nan	0.1000	-0.0001

Iter	TrainDeviance	ValidDeviance	StepSize	Improve
1	0.3487	-nan	0.1000	0.0188
2	0.3294	-nan	0.1000	0.0073
3	0.3138	-nan	0.1000	0.0062
4	0.3025	-nan	0.1000	0.0041
5	0.2928	-nan	0.1000	0.0034
6	0.2841	-nan	0.1000	0.0028
7	0.2759	-nan	0.1000	0.0029
8	0.2689	-nan	0.1000	0.0025
9	0.2618	-nan	0.1000	0.0024
10	0.2554	-nan	0.1000	0.0020
20	0.2103	-nan	0.1000	0.0011
40	0.1549	-nan	0.1000	0.0001
60	0.1228	-nan	0.1000	-0.0001
80	0.0991	-nan	0.1000	-0.0002
100	0.0816	-nan	0.1000	-0.0001
120	0.0670	-nan	0.1000	-0.0001
140	0.0576	-nan	0.1000	-0.0000
150	0.0534	-nan	0.1000	-0.0001

Iter	TrainDeviance	ValidDeviance	StepSize	Improve
1	0.3476	-nan	0.1000	0.0186
2	0.3269	-nan	0.1000	0.0090
3	0.3123	-nan	0.1000	0.0060
4	0.2992	-nan	0.1000	0.0056
5	0.2907	-nan	0.1000	0.0035
6	0.2818	-nan	0.1000	0.0035
7	0.2741	-nan	0.1000	0.0028
8	0.2661	-nan	0.1000	0.0026
9	0.2606	-nan	0.1000	0.0020
10	0.2544	-nan	0.1000	0.0022
20	0.2112	-nan	0.1000	0.0010
40	0.1558	-nan	0.1000	0.0002
60	0.1237	-nan	0.1000	0.0003
80	0.1025	-nan	0.1000	-0.0000
100	0.0863	-nan	0.1000	-0.0000
120	0.0740	-nan	0.1000	0.0000
140	0.0644	-nan	0.1000	-0.0001
150	0.0597	-nan	0.1000	-0.0000

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1705   37
         1    5   41

               Accuracy : 0.9765
                 95% CI : (0.9684, 0.983)
    No Information Rate : 0.9564
    P-Value [Acc > NIR] : 3.986e-06

                  Kappa : 0.65

 Mcnemar's Test P-Value : 1.724e-06

            Sensitivity : 0.9971
            Specificity : 0.5256
         Pos Pred Value : 0.9788
         Neg Pred Value : 0.8913
             Prevalence : 0.9564
         Detection Rate : 0.9536
   Detection Prevalence : 0.9743
      Balanced Accuracy : 0.7614

       'Positive' Class : 0
```

8. Conclusion

a. Using Random forest, we inferred that “Company_logo” is playing a significant role in identifying a job as fake or not. So companies should ensure to have a company logo while posting jobs.
b. Using Random forest, we inferred that “required_education” is also playing a significant role in identifying a job as fake or not. Students should ensure to strictly meet the education qualification while applying for the jobs.
c. Using logistic regression by adjusting the threshold we are able to get a high sensitivity of 95.37 for identifying the fake jobs.
d. Using Naïve Bayes, we are able to get a high sensitivity of 99.70% for identifying genuine jobs.
e. After applying text analytics, result predictions of all models indicated that this problem was a high sensitivity and low specificity problem. The RF algorithm seems to beat the others in terms of accuracy and sensitivity. SVM has the highest specificity. But looking at the amount of time spent on tuning the parameters, GBM and SVM were much faster compared to RF algorithms. Keeping accuracy and the simulation run-time in mind, I would choose the GBM to be the best.

9.Appendix

---

title: "Fake JobPosting Prediction"

author: "Priyank Prakash Babu"

date: "6/10/2020"

output: pdf_document

```{r setup, include=FALSE}

knitr::opts_chunk$set(echo = TRUE)

```

```{r loadpackages, include = FALSE}

if(!require("pacman")) install.packages("pacman")

pacman::p_load(e1071, ggplot2, caret, rmarkdown, corrplot,
tidyverse,"RCurl",dslabs,gridExtra,tm,"party",dpylr,stopwords,randomForest,readr, SnowballC, tictoc,
lubridate,data.table,wordcloud,grid,tinytex,tidytext,readr,"RColorBrewer",DMwR,gbm)

```

```{r loadData}

fake_job = read.csv("fake_job_postings.csv")

view(fake_job)

str(fake_job)

```

```{r Exploratory Data Analysis}

# List of unique columns

col_names <- colnames(fake_job)
```

```
#str(JobPosting_Data)
unique_col_count <-  fake_job %>%
  summarise(n_title = n_distinct(title),
            n_location = n_distinct(location),
            n_department = n_distinct(department),
            n_salary_range = n_distinct(salary_range),
            n_employment_type = n_distinct(employment_type),
            n_required_experience = n_distinct(required_experience),
            n_required_education = n_distinct(required_education),
            n_industry = n_distinct(industry),
            n_function = n_distinct(function.),
            n_fraudulent = n_distinct(fraudulent))

print(unique_col_count)
```

```{r}
# Distribution of jobs

fake_job %>% group_by(fraudulent) %>%  ggplot(aes(fraudulent, group = fraudulent)) +
  geom_bar(aes(fill = fraudulent), stat = "count") +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  geom_text(aes(label=..count..),stat='count',position=position_stack(vjust=0.5)) +
  ggtitle("Genuine Vs. Fraud Jobs") + xlab("Fradulent Flag") + ylab("Count of Job") + theme_bw()
```

```{r}
# Distribution of degrees
```

```
degree_distribution <- fake_job %>% group_by(required_education, fraudulent) %>% summarise(count = n())


degree_distribution %>%  ggplot(aes(reorder(
  degree_distribution$required_education, -degree_distribution$count), degree_distribution$count)) +
  geom_bar(stat = "identity", aes(fill = fraudulent)) +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ggtitle("Distribution of degrees") + xlab("Required Education") + ylab("Job Count")
```


```{r}
# Distribution of experience
experience_distribution <- fake_job %>% group_by(required_experience, fraudulent) %>% summarise(count = n())


experience_distribution %>%  ggplot(aes(reorder(
  experience_distribution$required_experience, -experience_distribution$count), experience_distribution$count)) +
  geom_bar(stat = "identity", aes(fill = fraudulent)) +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ggtitle("Jobs Per Required Experience Feature") + xlab("Required Experience") + ylab("Job Count")
```


```{r}
# Distribution of Employment Types
employment_type_distribution <- fake_job %>% group_by(employment_type, fraudulent) %>% summarise(count = n())


employment_type_distribution %>%  ggplot(aes(reorder(
```

```
  employment_type_distribution$employment_type, -employment_type_distribution$count),
employment_type_distribution$count)) +
  geom_bar(stat = "identity", aes(fill = fraudulent)) +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ggtitle("Jobs Per Required Employment Types Feature") + xlab("Employment Type") + ylab("Job Count")
```

```{r}
# Distribution of experience and education
fake_job %>% group_by(required_education) %>% ggplot(aes(x = required_education), group =
required_experience) +
  geom_bar(aes(fill = fake_job$required_experience), stat = "count") +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ggtitle("Jobs Per Required Education and Experience") + xlab("Required Education") +
  ylab("Job Count") + labs(fill='Required Experience')
```

```{r}
# Distribution of experience and employment type
fake_job %>% group_by(employment_type) %>% ggplot(aes(x = employment_type), group =
required_experience) +
  geom_bar(aes(fill = fake_job$required_experience), stat = "count") +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ggtitle("Jobs Per Required Experience") + xlab("Employment Type") +
  ylab("Job Count") + labs(fill='Required Experience')
```

```{r}
# Distribution of education and employment type
```

```
fake_job %>% group_by(employment_type) %>% ggplot(aes(x = employment_type), group = required_education) +
  geom_bar(aes(fill = fake_job$required_education), stat = "count") +
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ggtitle("Jobs Per Required Education") + xlab("Employment Type") +
  ylab("Job Count") + labs(fill='Education Level')
```

```{r}
levels(fake_job$benefits)<-c(levels(fake_job$benefits),"None")  #Add the extra level to your factor
fake_job$benefits[is.na(fake_job$benefits)] <- "None"

EmptyValues_df <- fake_job %>% summarise(Empty_JobIDs = sum(job_id == ""), Empty_Titles = sum(title == ""),
                                         Empty_Locations = sum(location == ""),
                                         Empty_Depts = sum(department == ""), Empty_SalRanges = sum(salary_range == ""),
                                         Empty_CompanyProfiles = sum(company_profile ==""), Empty_Desciptions = sum(description == ""),
                                         Empty_Requirements = sum(requirements == ""), Empty_Benefits = sum(benefits == ""),
                                         Empty_Telecommuting = sum(telecommuting ==""), Empty_HasLogo = sum(has_company_logo == ""),
                                         Empty_HasQuestions = sum(has_questions == ""),Empty_EmpType = sum(employment_type == ""),
                                         Empty_ReqExperience = sum(required_experience == ""), Empty_ReqEducation = sum(required_education ==""),
                                         Empty_Industry = sum(industry == ""), Empty_Function = sum(function. == ""),
                                         Empty_Fraudulent = sum(fraudulent == ""))
EmptyValues_df <- as.data.frame(t(EmptyValues_df))
EmptyValues_df$Names <- rownames(EmptyValues_df)
```

```
ggplot(EmptyValues_df, aes(x = EmptyValues_df$Names, y = EmptyValues_df$V1)) + geom_bar(stat = "Identity", fill = "blue")+
  theme(axis.text.x = element_text(angle = 90, size = 10)) +
  ggtitle("Empty Features") + xlab("Column Name") +
  ylab("Count")
```

```{r Modeling data}

str(fake_job)

fake.job<-fake_job[,c(1,10,11,12,13,14,15,16,17,18)]

fake.job$telecommuting<-as.factor(fake.job$telecommuting)
fake.job$has_company_logo<-as.factor(fake.job$has_company_logo)
fake.job$has_questions<-as.factor(fake.job$has_questions)
fake.job$fraudulent<-as.factor(fake.job$fraudulent)
fake.job$employment_type<-as.factor(fake.job$employment_type)
fake.job$required_experience<-as.factor(fake.job$required_experience)
fake.job$required_education<-as.factor(fake.job$required_education)
fake.job$industry<-as.factor(fake.job$industry)
fake.job$function.<-as.factor(fake.job$function.)

fake.job.impute<-fake.job[,c(1,2,3,4,5,6,7,10)]

data<-centralImputation(fake.job.impute)

sapply(data,function(x) sum(is.na(x)))
```

```
data$job_id<-NULL
set.seed(42)
train.index<-createDataPartition(data$fraudulent,p=0.8,list = FALSE)
train.df<-data[train.index,]
test.df<-data[-train.index,]
```

```{r Logistic Regression}
log.fake<-glm(fraudulent~.,data = train.df,family = "binomial")
pred<-predict(log.fake,test.df,type = "response")
#confusionMatrix(test.df$fraudulent,pred)
#pred<-predict(log.fake,test.df,type = "response")
#table(test.df$fraudulent , pred > 0.5)
table(test.df$fraudulent , pred > 0.02)
table(test.df$fraudulent , pred > 0.01)
```

```{r random forest}
randomForest.fake<-randomForest(fraudulent~.,data = train.df)
pred<-predict(randomForest.fake,test.df)
confusionMatrix(test.df$fraudulent,pred)
importance(randomForest.fake)
varImpPlot(randomForest.fake)
```

```{r Naive Bayes}
nb.fit = naiveBayes(fraudulent~., data = train.df)
preditions = predict(nb.fit, test.df, type = 'class')
```

```
confusionMatrix(preditions,test.df$fraudulent)
summary(nb.fit)
nb.fit
```

```{r SVM}
SVM.fake = gbm(fraudulent~., data = train.df, distribution = "gaussian",n.trees = 10, verbose = F)
prediction = predict(SVM.fake, test.df, n.tress = 10)
confusionMatrix(prediction,test.df$fraudulent)
```

```{r Modelling data with Text Analytics}
# Create the corpus object
corpus <- Corpus(VectorSource(fake_job$description))
# Remove punctuation
corpus <- tm_map(corpus, removePunctuation)
# Remove stop words
corpus <- tm_map(corpus, removeWords, stopwords(language = "en"))
# Perform stemmign
corpus <- tm_map(corpus, stemDocument)

# High frequency words using documenttermmatrix
frequencies <- DocumentTermMatrix(corpus)
# Removing sparse data
sparse_data <- removeSparseTerms(frequencies, 0.995)
# Converting to dataframe for furthur analysis
sparse_data_df <- as.data.frame(as.matrix(sparse_data))
```

```
# Assigning column names
colnames(sparse_data_df) <- make.names(colnames(sparse_data_df))
# Adding the dependent variable
sparse_data_df$fraudulent <- fake_job$fraudulent
# Removing duplicate column names
colnames(sparse_data_df) <- make.unique(colnames(sparse_data_df), sep = "_")

set.seed(2020, sample.kind = "Rounding")
test_index <- createDataPartition(y = sparse_data_df$fraudulent, times = 1, p = 0.1, list= FALSE)
train_set <- sparse_data_df[-test_index, ]
validation <- sparse_data_df[test_index, ]
train_set$fraudulent = as.factor(train_set$fraudulent)
validation$fraudulent = as.factor(validation$fraudulent)
```

```{r Naive Bayes}
nb.fit = naiveBayes(fraudulent~., data = train_set)
preditions = predict(nb.fit, validation, type = 'class')
mean(preditions == validation$fraudulent)

results = data.frame(Predicted=preditions, Actual=train_set[,'fraudulent'] )
confusionMatrix(preditions,validation$fraudulent)

```

```{r SVM}
n = 1000

b = 2
```

```
index <- sample(nrow(train_set), n)


svm_train_data <- train_set[index, ]


ctrl <- trainControl(method = "cv", verboseIter = FALSE, number = 5)


svm_fit <- train(fraudulent ~ ., data = svm_train_data, method = "svmLinear", preProcess =
c("center","scale"), trControl = ctrl)

#Tuning Parameter

grid_svm <- expand.grid(C = c(0.01, 0.1, 1, 10, 20))

#Training Model

svm_fit <- train(fraudulent ~ .,data = train_set, method = "svmLinear", preProcess = c("center","scale"),
tuneGrid = grid_svm, trControl = ctrl)


ctrl <- trainControl(method = "cv", verboseIter = TRUE, number = 5)

grid_svm <- expand.grid(C = c(0.01))

svm_fit <- train(fraudulent ~ .,data = train_set,

        method = "svmLinear", preProcess = c("center","scale"),

        tuneGrid = grid_svm, trControl = ctrl)


#Prediction

svm_pred <- predict(svm_fit, newdata = validation)

#Confusion Matrix

CM_svm <- confusionMatrix(svm_pred, validation$fraudulent)


CM_svm
```
```
```

```{r Random Forest}

control<- trainControl(method = "cv", number = 5, verboseIter = TRUE)

grid <-data.frame(mtry = c(100))


train_rf <- train(fraudulent ~ ., method = "rf", data = train_set, ntree = 150, trControl = control,tuneGrid = grid)


predict_rf <- predict(train_rf, newdata = validation)


confMatrix_rf <- confusionMatrix(predict_rf, validation$fraudulent)


confMatrix_rf

```



```{r Stochastic Gradient Boosting}

n <- 1000


control<- trainControl(method = "cv", number = 5)


gbmGrid <+- expand.grid(interaction.depth = c(1, 5, 9), n.trees = (1:30)*50, shrinkage = 0.1, n.minobsinnode = 20)


index <- sample(nrow(train_set), n)


gbm_train_data <- train_set[index, ]


subset_train_gbm <- train(fraudulent ~ ., method = "gbm", data = gbm_train_data, trControl = control, verbose = TRUE, tuneGrid = gbmGrid)
```

```
gbmGrid <- expand.grid(interaction.depth = c(1, 5, 10, 25, 50, 100), n.trees = 150, shrinkage = 0.1, n.minobsinnode = 20)

train_gbm <- train(fraudulent ~ ., method = "gbm", data = train_set, trControl = control, verbose = TRUE, tuneGrid = gbmGrid)
```

10. References :

https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

https://www.kaggle.com/

https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

https://statweb.stanford.edu/~jhf/ftp/stobst.pdf

https://www.sciencedirect.com/science/article/abs/pii/S0167947301000652

https://en.wikipedia.org/wiki/Gradient_boosting