Major Project Report on

Classifying fraudulent companies using ML Algorithm in Python

Submitted By

Sonali Gupta

Roll no: 2K19/DMBA/097

Under the Guidance of

Dr. Sonal Thukral

Assistant Professor

DELHI SCHOOL OF MANAGEMENT

Delhi Technological University

Bawana Road Delhi 110042

CERTIFICATE

This is to certify that the work titled 'Classifying fraudulent companies using ML Algorithm in Python' as part of the final year Major Research Project submitted by Sonali Gupta in the 4th Semester of MBA, Delhi School of Management, Delhi Technological University during January-May 2021 is her original work and has not been submitted anywhere else for the award of any credits/ degree whatsoever.

The project is submitted to Delhi School of Management, Delhi Technological University in partial fulfilment of the requirement for the award of the degree of Master of Business Administration.

ACKNOWLEDGEMENT

It is a great pleasure for me to acknowledge the kind of help and guidance received during the research work. I would like to thank my faculty advisor Dr.Sonal Thukral who helped me to take up the topic 'Classifying fraudulent companies using ML Algorithm in Python' and guided me to complete this project properly.

I am highly indebted to Delhi School of Management, Delhi Technological University for giving me an opportunity to work on this project.

Lastly, I would like to express my gratitude to all the honorable faculty members for sharing their experience and expertise on this project. I have put all my efforts to ensure that the project is completed in the best possible manner and also ensured that the project is error-free.

Sonali Gupta

2K19/DMBA/097

ABSTRACT

This paper is a case study of visiting an external audit company to explore the usefulness of machine learning algorithms for improving the quality of an audit work. Annual data of 777 firms from 14 different sectors are collected.

With the appearance of tremendous growth of financial fraud cases, machine learning will play a big part in improving the quality of an audit field work in the future

Purpose: The goal of the research is to help the auditors by building a classification model that can predict the fraudulent firm on the basis of the present risk factors and historical risk factors. The information about the sectors and the counts of firms are listed respectively as Irrigation (114), Public Health (77), Buildings and Roads (82), Forest (70), Corporate (47), Animal Husbandry (95), Communication (1), Electrical (4), Land (5), Science and Technology (3), Tourism (1), Fisheries (41), Industries (37), Agriculture (200).

Methodology/Approach: The machine learning algorithms like Random Forest Classifier and Logistic regression are used in this project to classify the fraudulent firms.The exploratory data analysis is done using libraries of Python like matplotlib and plotly.

Research Limitations: The dataset is one year non-confidential data in the year 2015 to 2016 of firms is collected from the Auditor Office of India to build a predictor for classifying suspicious firms.

Value: To help the auditors by building a classification model that can predict the fraudulent firm on the basis the present and historical risk factors.

TABLE OF CONTENTS

1. Introduction

Fraud is a critical issue worldwide. Firms that resort to the unfair practices without the fear of legal repercussion have a grievous consequence on the economy and individuals in the society. Auditing practices are responsible for fraud detection. Audit is defined as the process of examining the financial records of any business to corroborate that their financial statements are in compliance with the standard accounting laws and principles (Cosserat 2009). It is a very exacting task to detect firms in spotting frauds, detecting errors, and disclosing employees guilty of abetting illegal transactions. Data analytics tools for an effective fraud management have become the need of the hour for an audit.

An audit is performed usually by an independent body especially onsite to inspect or verify different aspects of a corporation such as financial records, quality control, etc. to ensure everything is done according to proper guidelines. Audits can be performed on entire organizations activities or it can remain focused on a specific function. One of the main objectives of an audit is to find discrepancies and faults in data provided by corporations and building a machine learning model can help this cause. A typical audit has a audit cycle which the auditors follow to execute these audits in a systematic manner. Therefore introducing an audit process to an automated system can help speed up things which can usually take days to implement. Automating an audit process should be the goal of an auditor as to make the audit process more efficient. As more and more audits are going on a process to become automated, classifying and predicting discrepancies during a companies activities while maintaining accuracy as a desired objective. Many attributes are analyzed during the working of machine learning model but one attribute that is helpful in predicting this faults in these companies is Risk factor. Risk factors are calculated on many other circumstantial attributes present in the dataset.

Annual data of 777 firms from 14 different sectors are collected. With the appearance of tremendous growth of financial fraud cases, machine learning will play a big part in improving the quality of an audit field work in the future.
In this paper I will be using Random forest classifier as my machine learning algorithm to classify the fraudulent companies out of 777 firms
The purpose of classifying the firms is to maximize the field-testing work of high-risk firms that warrant significant investigation

2. Literature Review

Generally, audits are classified into two categories as internal and external auditing (Cosserat 2009). Internal-audit, although is an independent department of an organization, but resides within the organization. These are company-employees who are accountable for performing audits of financial and nonfinancial statements as per their annual audit plan. External audit is a fair and independent regular audit authority, which is responsible for an annual statutory audit of financial records. The external audit company has a fiduciary duty and is critical to the proper conduct of business. For instance, their work is to audit the receipts, expenditures, accounts related to the trading, profit, contingency funds, balance sheets, public accounts, etc. kept in any government office. It is their duty to ensure that the funds allocated to any government department have been put to use as per law. On successful completion of an audit process, auditors deliver an audit and inspection summary report called audit paras to the company comprising of the details of all the findings from the audit. This may include discrepancies, noncompliance of accounting rules, leakage of revenue, inaccurate calculations, etc (Tschakert 2016)

Auditing Standards Board Task Force (ASBTF) is also working on developing an innovative Audit Data Analytics Guide in order to integrate the data analytics tools for the auditing tasks (Maria, Murphy and Tysiac 2015).

Machine learning has got much attentiveness in the data analytics as it offers new computational as well as epistemological techniques to produce better results. Machine learning proposes several algorithms that are derived from the area of statistics and artificial intelligence. Many researchers have employed algorithms like artificial neural network, logistic regression, decision trees, and Bayesian belief networks for detecting management fraud in the financial statements (Fanning and Cogger 1998; Green and Choi 1997; Spathis 2002). The ensemble machine learning method is also applied successfully for improving the classification accuracies of the auditing task (Kotsiantis 2006). Machine learning algorithms like support vector machine, logistic regression, probabilistic neural network, genetic algorithm, etc. are also combined with feature selection methods in order to prove their usability in detecting fraud in the Chinese firms (Ravisankar 2011). In a review of data analytics tools like for fraud prediction, clustering, and outlier detection that are used for fraud management task, researchers listed algorithms like neural network, decision tree, Bayesian network, etc. as most commonly used methods (Sharma 2013).

The prime goal of an auditor during an audit-planning phase is to follow a proper analytical procedure to impartially and appropriately identify the firms that resort to high risk of unfair practices. Predictive analytics is also implemented using machine learning methods because it provides actionable insights for the audit companies. One of the most common applications of predictive analytics in audit is the classification of suspicious firm. Identifying fraudulent firms can be studied as a classification problem. The purpose of classifying the firms during the preliminary stage of an audit is to maximize the field-testing work of high-risk firms that warrant significant investigation. According to a research, data analytics has benefited internal auditing

more as compared to advancements it has contributed for the external audits (Tysiac 2015)

On successful completion of an audit process, auditors deliver an audit and inspection summary report called audit paras to the company comprising of the details of all the findings from the audit. This may include discrepancies, noncompliance of accounting rules, leakage of revenue, inaccurate calculations, etc. The whole audit process flow is summarized in Figure 1.

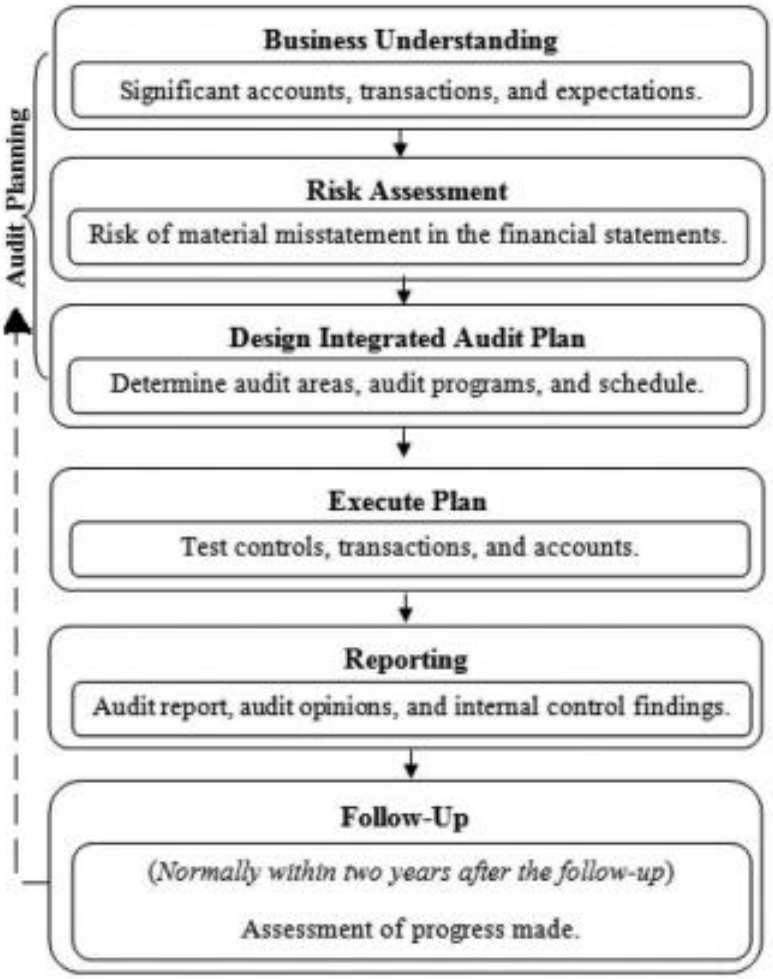


Figure 1. Audit work-flow.

3. Objectives

The goal of the project is to help the auditors by building a classification model that can predict the fraudulent firm on the basis of the present risk factors and historical risk factors.

4. Motivation of the Study

- The major motivation was that I wanted to study about a dataset which contains audit data as I worked as an Internal Audit Intern and wanted to amalgamate my knowledge of both data analysis and Audit.
- Applying **machine learning** to **fraud** detection enables audit **firms to identify** genuine activities versus **fraudulent** activities in real time, and with greater accuracy so wanted to create classification model which can be used by the auditing firms

5. Research Methodology

5.1 Data Set

- Exhaustive one year non-confidential data in the year 2015 to 2016 of firms is collected from the Auditor Office of India to build a predictor for classifying suspicious firms.
- I wanted to do study on the audit data and the most reliable source which was a government website mentioned below had the latest dataset of 2015-2016.
- There are total 777 firms data from 46 different cities of a state that are listed by the auditors for targeting the next field-audit work. The target-offices are listed from 14 different sectors.
- Many risk factors are examined from various areas like past records of audit office, audit-paras, environmental conditions reports, firm reputation summary, on-going issues report, profit-value records, loss-value records, follow-up reports etc. After in-depth interview with the auditors, important risk factors are evaluated and their probability of existence is calculated from the present and past records.
- The dataset was made accessible through a pair of csv files named as – audit_risk and trial.
- The audit_risk.csv contains 27 columns in total and trial.csv contains 18 columns in total. E.g. Sector_Score, LOCATION_ID, etc. are the contents of the two csv files.
- From all of these columns in this dataset, two main columns (PARA_A, PARA_B) are vital in risk calculation. Each ofthese columns contain inconsistencies found in planned and unplanned expenses
- **Source**: https://archive.ics.uci.edu/ml/datasets/Audit+Data
- The information about the sectors and the counts of firms are listed respectively as Irrigation (114), Public Health (77), Buildings and Roads (82), Forest (70), Corporate (47), Animal Husbandry (95), Communication (1), Electrical (4), Land (5), Science and Technology (3), Tourism (1), Fisheries (41), Industries (37), Agriculture (200).

- The dataset has these major attributes which are as follows
  a. **Sector_score**: Historical risk score value of the target-unit using analytical procedure
  b. **LOCATION_ID**: Unique ID of the city/province.
  c. **PARA_A**: Discrepancy found in the planned expenditure of inspection and summary report A in Rs (in crore)
  d. **PARA_B**: discrepancy found in the plannedexpenditure of inspection and summary report B in Rs (in crore)
  e. **Total**: Total amount of discrepancy found in other reports Rs (in crore).
  f. **Number**: Historical discrepancy score.

5.2 Algorithms Used

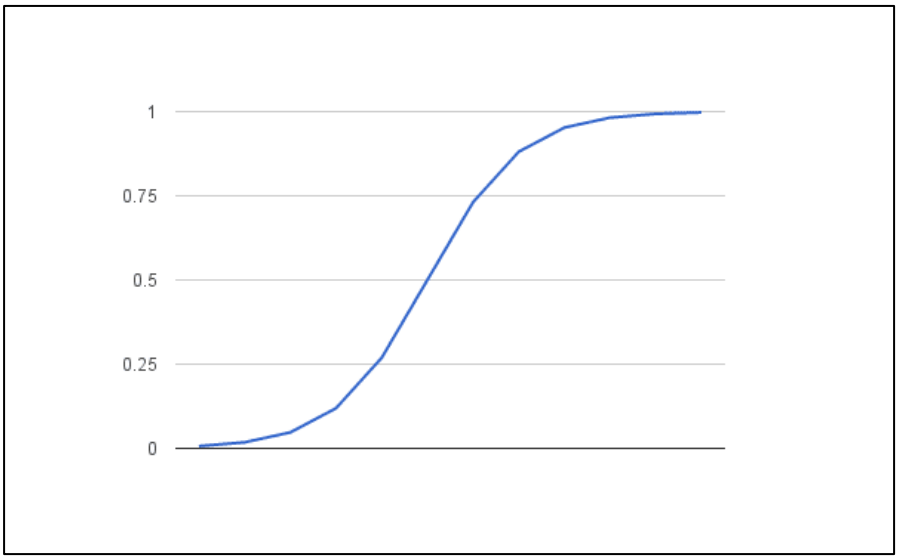
a. <u>Logistic Regression</u>

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.



Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

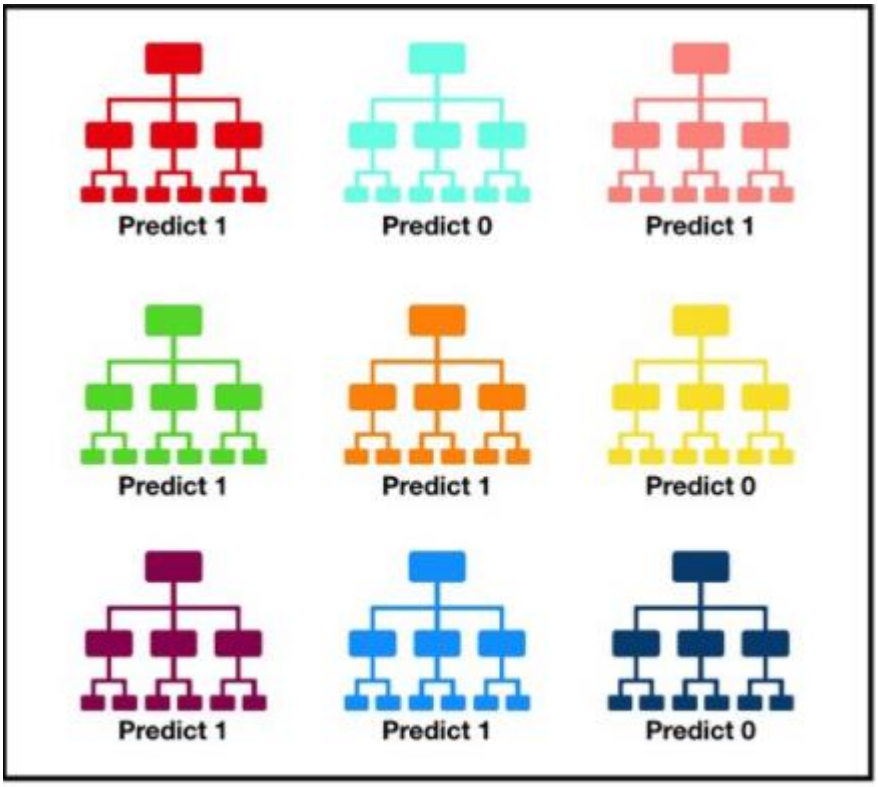
Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

b. <u>Random Forest</u>

A big part of machine learning is classification — we want to know what class (a.k.a. group) an observation belongs to. The ability to precisely classify observations is extremely valuable for various business applications like predicting whether a particular user will buy a product or forecasting whether a given loan will default or not.
Data science provides a plethora of classification algorithms such as logistic regression, support vector machine, naive Bayes classifier, and decision trees. But near the top of the classifier hierarchy is the random forest classifier

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction



Tally: Six 1s and Three 0s
Prediction: 1

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:
A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:
There needs to be some actual signal in our features so that models built using those features do better than random guessing.
The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

5.3 Programming Tool Used

The programming tool used is Python 3.0 using Jupyter Notebooks.

Jupyter Notebooks

The Jupyter Notebook is an application available on web which is open source and that permits you to make and share archives that contain live code, conditions, representations and account text. Utilizations include: information cleaning and change, mathematical re-enactment, measurable displaying, information representation, AI, and substantially more. It utilizes a scratch pad which incorporates code and its yield into a solitary record that consolidates representations, story text, numerical conditions, and other rich media. All in all: it's a solitary report where you can run code, show the yield, and furthermore add clarifications, equations, diagrams, and make your work more straightforward, reasonable, repeatable, and shareable.

5.4 Libraries used

- **Pandas**

  Pandas is a Python bundle that gives quick, adaptable, and expressive information structures intended to make working with organized (plain, multidimensional, conceivably heterogeneous) and time arrangement information both simple and natural. This library will be utilized to import the source of data and make frame of data for examination

.

- **Sklearn**

  Sklearn is the most valuable and strong library for ML in Python. It gives a determination of effective devices for ML and measurable demonstrating including grouping, relapse, bunching and dimensionality decrease through a consistence interface in Python. This library, which is to a great extent written in Python, is based upon NumPy, SciPy and Matplotlib. Sklearn will be utilized to apply arbitrary timberland regressor for forecast and figure the precision of the model.

- **PlotLy**

  The plotly is an interactive library of python language and is an intelligent, open-source plotting library that upholds more than 40 special outline types covering a wide scope of measurable, monetary, geographic, logical, and 3-dimensional use-cases. plotly empowers Python clients to make wonderful intuitive electronic representations that can be shown in Jupyter scratch pad, saved to independent HTML documents, or filled in as a feature of unadulterated Python-fabricated web applications utilizing Dash.

6. Code and Analysis

6.1 Data Preprocessing

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, lacking in certain behaviors or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing. Data pre-processing is used in database-driven applications such as customer relationship management and rule-based applications (like neural networks).

- **Import Libraries**

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns

%matplotlib inline
import itertools
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import  accuracy_score, f1_score, precision_score,confusion_ma
trix, recall_score, roc_auc_score
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from sklearn.svm import SVC
plt.style.use('fivethirtyeight')
```

- **Data Understanding**

```
df=pd.read_csv('audit_data.csv')
df.head()
```

	Sector_score	LOCATION_ID	PARA_A	Score_A	Risk_A	PARA_B	Score_B	Risk_B	TOTAL	numbers
0	3.89	23	4.18	0.6	2.508	2.50	0.2	0.500	6.68	5.0
1	3.89	6	0.00	0.2	0.000	4.83	0.2	0.966	4.83	5.0
2	3.89	6	0.51	0.2	0.102	0.23	0.2	0.046	0.74	5.0
3	3.89	6	0.00	0.2	0.000	10.80	0.6	6.480	10.80	6.0
4	3.89	6	0.00	0.2	0.000	0.08	0.2	0.016	0.08	5.0

```
df.columns
```

```
Index(['Sector_score', 'LOCATION_ID', 'PARA_A', 'Score_A', 'Risk_A', 'PA
RA_B',
       'Score_B', 'Risk_B', 'TOTAL', 'numbers', 'Score_B.1', 'Risk_C',
       'Money_Value', 'Score_MV', 'Risk_D', 'District_Loss', 'PROB', 'Ri
Sk_E',
       'History', 'Prob', 'Risk_F', 'Score', 'Inherent_Risk', 'CONTROL_R
ISK',
       'Detection_Risk', 'Audit_Risk', 'Risk'],
      dtype='object')
```

df.tail()

	Sector_score	LOCATION_ID	PARA_A	Score_A	Risk_A	PARA_B	Score_B	Risk_B	TOTAL	numbers
771	55.57	9	0.49	0.2	0.098	0.40	0.2	0.080	0.89	5.0
772	55.57	16	0.47	0.2	0.094	0.37	0.2	0.074	0.84	5.0
773	55.57	14	0.24	0.2	0.048	0.04	0.2	0.008	0.28	5.0
774	55.57	18	0.20	0.2	0.040	0.00	0.2	0.000	0.20	5.0
775	55.57	15	0.00	0.2	0.000	0.00	0.2	0.000	0.00	5.0

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 776 entries, 0 to 775
Data columns (total 25 columns):
Sector_score      776 non-null float64
PARA_A            776 non-null float64
Score_A           776 non-null float64
Risk_A            776 non-null float64
PARA_B            776 non-null float64
Score_B           776 non-null float64
Risk_B            776 non-null float64
numbers           776 non-null float64
Score_B.1         776 non-null float64
Risk_C            776 non-null float64
Money_Value       775 non-null float64
Score_MV          776 non-null float64
Risk_D            776 non-null float64
District_Loss     776 non-null int64
PROB              776 non-null float64
RiSk_E            776 non-null float64
History           776 non-null int64
Prob              776 non-null float64
Risk_F            776 non-null float64
Score             776 non-null float64
Inherent_Risk     776 non-null float64
CONTROL_RISK      776 non-null float64
Detection_Risk    776 non-null float64
Audit_Risk        776 non-null float64
Risk              776 non-null int64
dtypes: float64(22), int64(3)
memory usage: 151.6 KB
```

df.describe()

	Sector_score	PARA_A	Score_A	Risk_A	PARA_B	Score_B	Risk_B	numbers
count	776.000000	776.000000	776.000000	776.000000	776.000000	776.000000	776.000000	776.000000
mean	20.184536	2.450194	0.351289	1.351029	10.799988	0.313144	6.334008	5.067655
std	24.319017	5.678870	0.174055	3.440447	50.083624	0.169804	30.072845	0.264449
min	1.850000	0.000000	0.200000	0.000000	0.000000	0.200000	0.000000	5.000000
25%	2.370000	0.210000	0.200000	0.042000	0.000000	0.200000	0.000000	5.000000
50%	3.890000	0.875000	0.200000	0.175000	0.405000	0.200000	0.081000	5.000000
75%	55.570000	2.480000	0.600000	1.488000	4.160000	0.400000	1.840500	5.000000
max	59.850000	85.000000	0.600000	51.000000	1264.630000	0.600000	758.778000	9.000000

- **Data Cleaning**

  df.isna().sum()

```
Sector_score      0
PARA_A            0
Score_A           0
Risk_A            0
PARA_B            0
Score_B           0
Risk_B            0
numbers           0
Score_B.1         0
Risk_C            0
Money_Value       1
Score_MV          0
Risk_D            0
District_Loss     0
PROB              0
RiSk_E            0
History           0
Prob              0
Risk_F            0
Score             0
Inherent_Risk     0
CONTROL_RISK      0
Detection_Risk    0
Audit_Risk        0
Risk              0
dtype: int64
```

  Observation: As we can see that Money_Value has one null value so I am imputing it with the mean value of this feature.

  df['Money_Value'].fillna((df['Money_Value'].mean()), inplace=True)

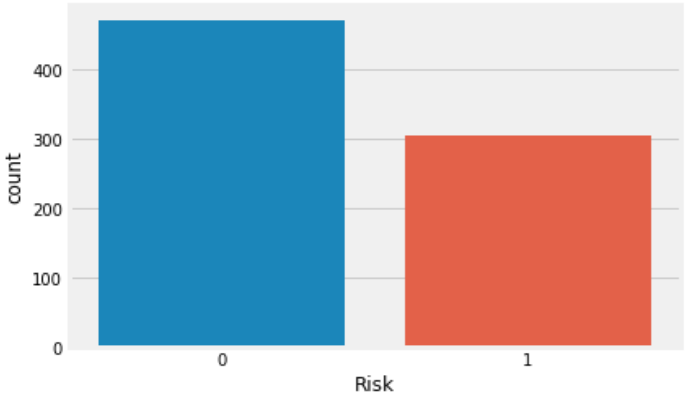
df.isna().sum()

```
Sector_score      0
PARA_A            0
Score_A           0
Risk_A            0
PARA_B            0
Score_B           0
Risk_B            0
numbers           0
Score_B.1         0
Risk_C            0
Money_Value       0
Score_MV          0
Risk_D            0
District_Loss     0
PROB              0
RiSk_E            0
History           0
Prob              0
Risk_F            0
Score             0
Inherent_Risk     0
CONTROL_RISK      0
Detection_Risk    0
Audit_Risk        0
Risk              0
```

6.2 Exploratory Data Analysis (EDA)

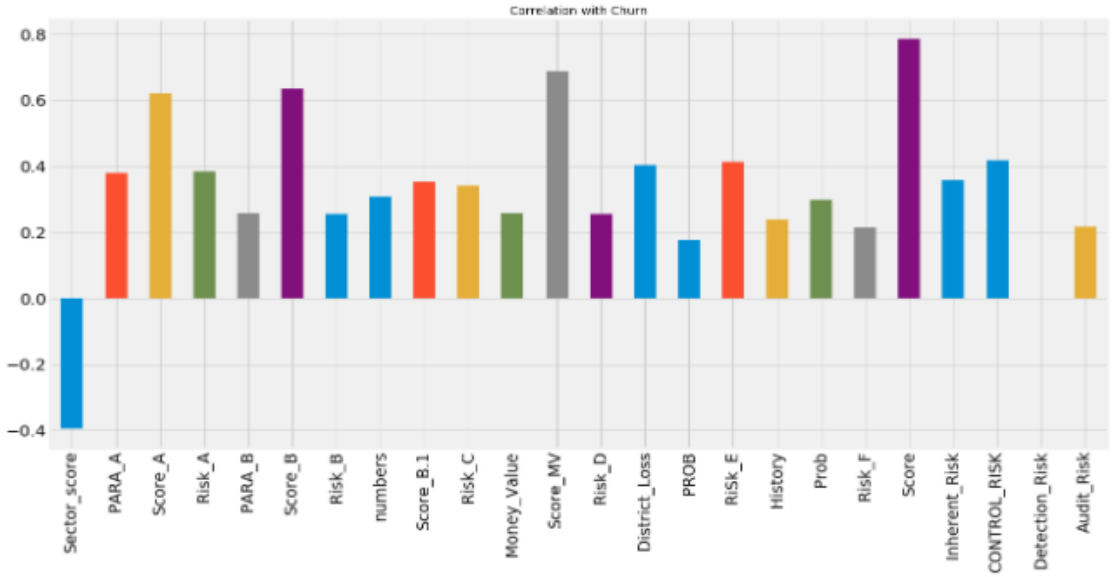
It is an approach to analysing datasets to summarize their main characteristics, often w ith visual methods. EDA is used for seeing what the data can tell us before the modelli ng task. It is not easy to look at a column of numbers or a whole spreadsheet and deter mine important characteristics of the data. It may be tedious, boring, and/or overwhel ming to derive insights by looking at plain numbers. Exploratory data analysis techniq ues have been devised as an aid in this situation. Exploratory data analysis is generally cross-classified in two ways. First, each method is either non-graphical or graphical. A nd second, each method is either univariate or multivariate (usually just bivariate).

```
sns.countplot(df['Risk'], label = "Count")
```




```
X=df.drop(['Risk'],axis=1)
```

```
X.corrwith(df.Risk).plot.bar(
    figsize = (20, 10), title = "Correlation with Churn", fontsize = 20,
    rot = 90, grid = True)
```




```
X.corr(method='pearson').style.format("{:.2}").background_gradient(cmap=plt.get_cmap('coolwarm'), axis=1)
```

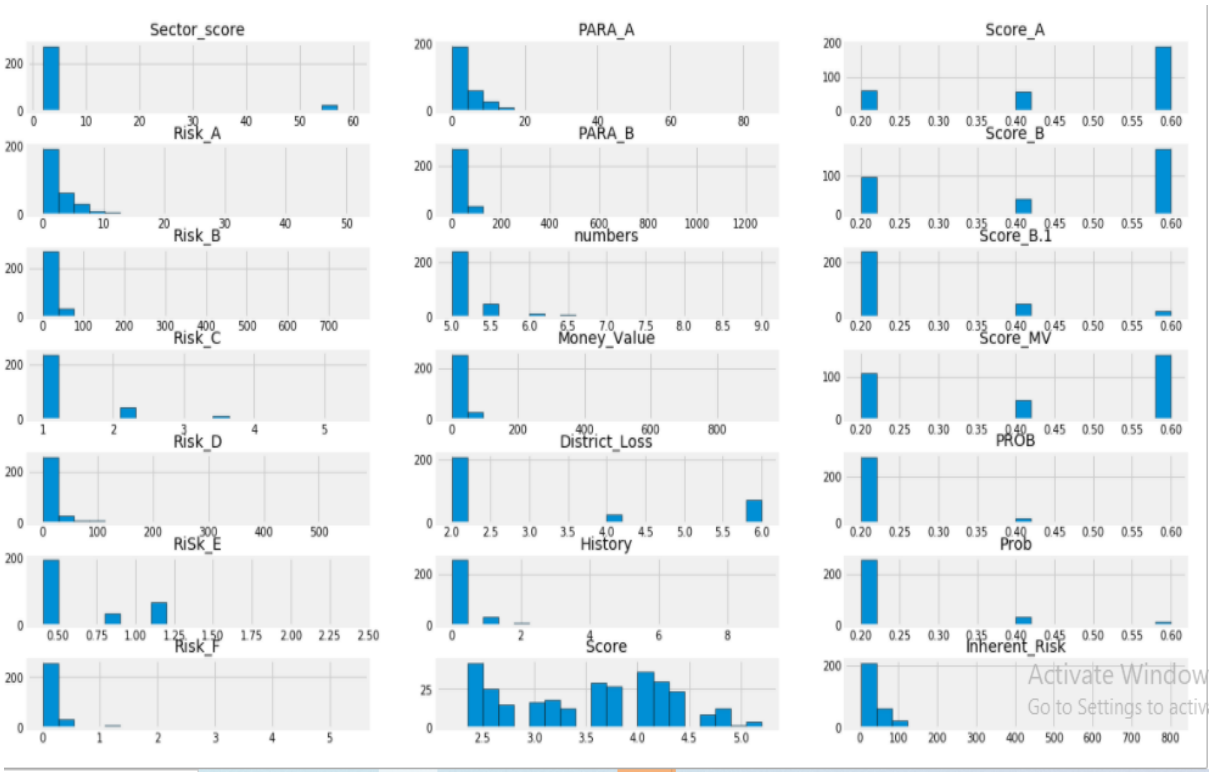
Sector_score	1.0	-0.22	-0.43	-0.22	-0.13	-0.22	-0.13	-0.15	-0.17	-0.17	-0.12
PARA_A	-0.22	1.0	0.5	1.0	0.16	0.36	0.16	0.13	0.14	0.14	0.45
Score_A	-0.43	0.5	1.0	0.5	0.25	0.57	0.25	0.24	0.27	0.27	0.21
Risk_A	-0.22	1.0	0.5	1.0	0.17	0.36	0.17	0.14	0.14	0.14	0.45
PARA_B	-0.13	0.16	0.25	0.17	1.0	0.35	1.0	0.21	0.23	0.22	0.13
Score_B	-0.22	0.36	0.57	0.36	0.35	1.0	0.35	0.28	0.31	0.3	0.21
Risk_B	-0.13	0.16	0.25	0.17	1.0	0.35	1.0	0.21	0.23	0.22	0.13
numbers	-0.15	0.13	0.24	0.14	0.21	0.28	0.21	1.0	0.91	0.96	0.19
Score_B.1	-0.17	0.14	0.27	0.14	0.23	0.31	0.23	0.91	1.0	0.99	0.22
Risk_C	-0.17	0.14	0.27	0.14	0.22	0.3	0.22	0.96	0.99	1.0	0.22
Money_Value	-0.12	0.45	0.21	0.45	0.13	0.21	0.13	0.19	0.22	0.22	1.0
Score_MV	-0.32	0.29	0.48	0.29	0.31	0.57	0.31	0.45	0.51	0.49	0.39
Risk_D	-0.12	0.45	0.2	0.45	0.12	0.2	0.12	0.19	0.22	0.22	1.0
District_Loss	-0.11	0.13	0.089	0.13	0.083	-0.0047	0.083	0.13	0.15	0.15	0.028
PROB	-0.087	0.044	0.094	0.044	0.043	0.093	0.043	0.036	0.037	0.036	0.032
RiSk_E	-0.13	0.12	0.1	0.12	0.079	0.015	0.08	0.14	0.16	0.15	0.033
History	-0.11	0.12	0.18	0.12	0.2	0.2	0.2	0.2	0.23	0.22	0.08
Prob	-0.14	0.17	0.27	0.18	0.32	0.31	0.32	0.21	0.25	0.24	0.11
Risk_F	-0.1	0.1	0.15	0.11	0.2	0.17	0.2	0.2	0.22	0.22	0.07
Score	-0.34	0.43	0.72	0.43	0.4	0.9	0.4	0.5	0.57	0.55	0.29
Inherent_Risk	-0.17	0.48	0.32	0.48	0.65	0.37	0.65	0.27	0.31	0.3	0.83
CONTROL_RISK	-0.15	0.15	0.17	0.15	0.19	0.13	0.19	0.23	0.26	0.25	0.07
Detection_Risk	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan

Observation: As we have seen there is no correlation of Detection_Risk so it will be deleted before model building process as it is not contributing well enough for prediction process.

```
X=X.drop(['Detection_Risk'],axis=1)
X.columns
```

```
Index(['Sector_score', 'PARA_A', 'Score_A', 'Risk_A', 'PARA_B', 'Score_B',
       'Risk_B', 'numbers', 'Score_B.1', 'Risk_C', 'Money_Value', 'Score_MV',
       'Risk_D', 'District_Loss', 'PROB', 'RiSk_E', 'History', 'Prob',
       'Risk_F', 'Score', 'Inherent_Risk', 'CONTROL_RISK', 'Audit_Risk'],
      dtype='object')
```

```
df1=df[df['Risk']==1]
columns=df1.columns[:21]
plt.subplots(figsize=(18,15))
length=len(columns)
for i,j in itertools.zip_longest(columns,range(length)):
    plt.subplot((length/2),3,j+1)
    plt.subplots_adjust(wspace=0.2,hspace=0.5)
    df1[i].hist(bins=20,edgecolor='black')
    plt.title(i)
plt.show()
```




```
y=df['Risk']
```

6.3 Train Test Split

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

```
from sklearn.model_selection import train_test_split,cross_val_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,stratify=y, rando
m_state = 123)
```

6.4 Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train_scaled = pd.DataFrame(sc_X.fit_transform(X_train))
X_test_scaled = pd.DataFrame(sc_X.transform(X_test))
```

6.5 Applying Base Model : Logistic Regression

```
logi = LogisticRegression(random_state = 0, penalty = 'l1')
logi.fit(X_train_scaled, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l1',
                   random_state=0, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

6.6 Cross Validation

Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. Use cross-validation to detect overfitting, ie, failing

to generalize a pattern

```
kfold = model_selection.KFold(n_splits=10, random_state=7)
scoring = 'accuracy'

acc_logi = cross_val_score(estimator = logi, X = X_train_scaled, y = y_train, cv = kfo
ld,scoring=scoring)
  acc_logi.mean()
```

```
0.9774193548387096
```

6.7 Model Evaluation

Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data.

Methods for evaluating a model's performance are divided into 2 categories: namely, holdout and Cross-validation. Both methods use a test set (i.e data not seen by the model) to evaluate model performance.

```
y_predict_logi = logi.predict(X_test_scaled)
acc= accuracy_score(y_test, y_predict_logi)
roc=roc_auc_score(y_test, y_predict_logi)
prec = precision_score(y_test, y_predict_logi)
rec = recall_score(y_test, y_predict_logi)
f1 = f1_score(y_test, y_predict_logi)

results = pd.DataFrame([['Logistic Regression',acc, acc_logi.mean(),prec,rec, f1,roc]],
            columns = ['Model', 'Accuracy','Cross Val Accuracy', 'Precision', 'Recall', 'F1 Sc
ore','ROC'])
results
```

	Model	Accuracy	Cross Val Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.980769	0.977419	0.967742	0.983607	0.97561	0.981277

Result:

The accuracy of Logistic Regression is 98%

6.8 Applying Random Forest¶

```
random_forest_e = RandomForestClassifier(n_estimators = 100,criterion='entropy', ra
ndom_state = 47)
random_forest_e.fit(X_train_scaled, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=47, verbose=0,
                       warm_start=False)
```

6.9 Cross Validation

```
acc_rande = cross_val_score(estimator = random_forest_e, X = X_train_scaled, y = y
_train, cv = kfold, scoring=scoring)
acc_rande.mean()
```

```
0.9983870967741936
```

6.10 Model evaluation

```
y_predict_r = random_forest_e.predict(X_test_scaled)
roc=roc_auc_score(y_test, y_predict_r)
acc = accuracy_score(y_test, y_predict_r)
prec = precision_score(y_test, y_predict_r)
rec = recall_score(y_test, y_predict_r)
f1 = f1_score(y_test, y_predict_r)

model_results = pd.DataFrame([['Random Forest',acc, acc_rande.mean(),prec,rec, f1,r
oc]],
             columns = ['Model', 'Accuracy','Cross Val Accuracy', 'Precision', 'Recall', 'F1
Score','ROC'])
results = results.append(model_results, ignore_index = True)
results
```

Result:

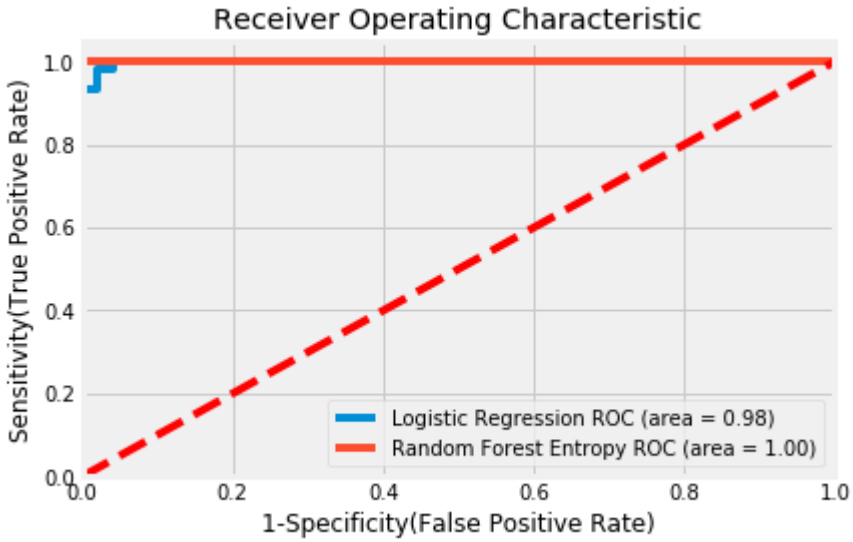
	Model	Accuracy	Cross Val Accuracy	Precision	Recall	F1 Score	ROC
0	Logistic Regression	0.980769	0.977419	0.967742	0.983607	0.97561	0.981277
1	Random Forest	1.000000	0.998387	1.000000	1.000000	1.00000	1.000000

5.11 Plotting ROC AUC Curve

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s

```
from sklearn import metrics
import matplotlib.pyplot as plt

plt.figure()

# Add the models to the list that you want to view on the ROC plot
models = [
{
    'label': 'Logistic Regression',
    'model': LogisticRegression(random_state = 0, penalty = 'l1'),
},
    {
    'label': 'Random Forest Entropy',
    'model': RandomForestClassifier(n_estimators = 100,criterion='entropy', random_stat
e = 47),
},

]

# Below for loop iterates through your models list
for m in models:
    model = m['model'] # select the model
    model.fit(X_train_scaled, y_train) # train the model
    y_pred=model.predict(X_test_scaled) # predict the test data
# Compute False postive rate, and True positive rate
    fpr, tpr, thresholds = metrics.roc_curve(y_test, model.predict_proba(X_test_scaled)[:,
1])
# Calculate Area under the curve to display on the plot
    auc = metrics.roc_auc_score(y_test,model.predict(X_test_scaled))
# Now, plot the computed values
    plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % (m['label'], auc))
# Custom settings for the plot
      plt.plot([0, 1], [0, 1],'r--')
```

```
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1-Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



Results:

- **Logistic Regression**

  Sensitivity=(TP/TP+FN) = (60/60+1)=0.98
  Specificity=(TN/TN+FP)=(93/93+2)=0.97
  1-Specificity=0.03

- **Random Forest Classifier**

  Sensitivity=(TP/TP+FN) = (61/61+0)= 1
  Specificity=(TN/TN+FP)=(95/95)=1
  1-Specificity=0

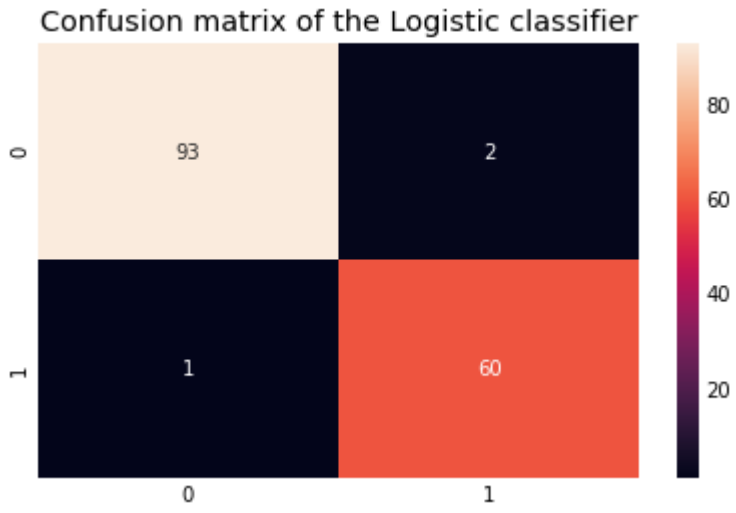
5.12 Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classif ication model (or "classifier") on a set of test data for which the true values are known The confusion matrix itself is relatively simple to understand, but the related terminolo gy can be confusing

Let's now define the most basic terms, which are whole numbers (not rates):

- true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.
- true negatives (TN): We predicted no, and they don't have the disease.
- false positives (FP): We predicted yes, but they don't actually have the disease. (Also k nown as a "Type I error.")
- false negatives (FN): We predicted no, but they actually do have the disease. (Also kn own as a "Type II error.")

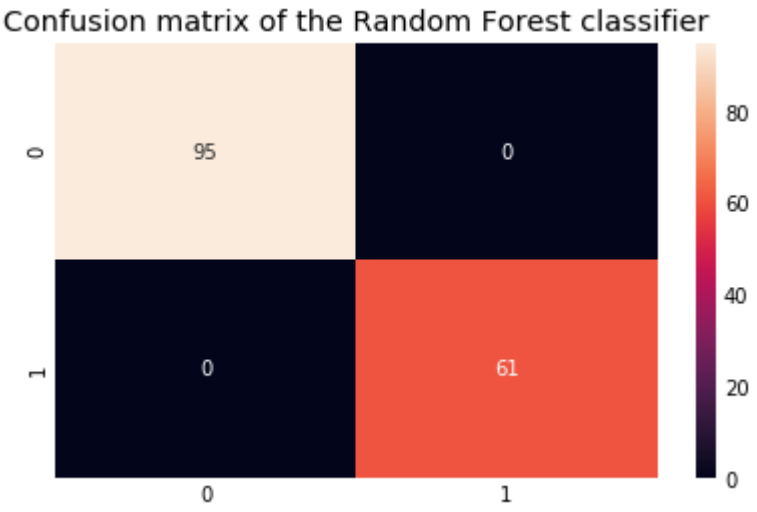
```
cm_logi = confusion_matrix(y_test, y_predict_logi)
plt.title('Confusion matrix of the Logistic classifier')
sns.heatmap(cm_logi,annot=True,fmt="d")
plt.show()
```



Result:

- **There are 93 firms true negative and 60 true positive**
- **There is 1 false negative and 2 false positive**

```
cm_r = confusion_matrix(y_test, y_predict_r)
plt.title('Confusion matrix of the Random Forest classifier')
sns.heatmap(cm_r,annot=True,fmt="d")
plt.show()
```

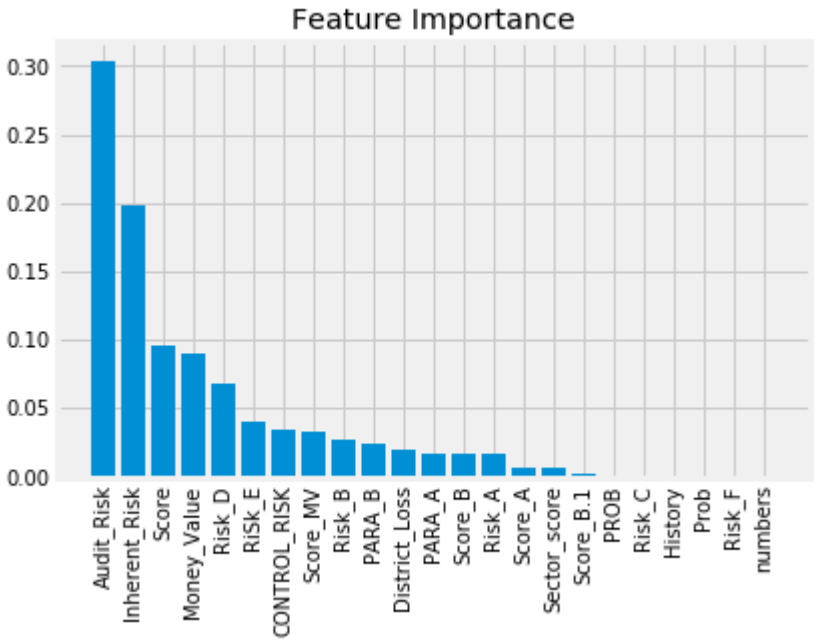


Result:

- **There are 95 firms true negative and 61 true positive**
- **There is 0 false negative and 0 false positive**

5.13 Feature Importance Plot

```
importances = random_forest_e.feature_importances_
indices = np.argsort(importances)[::-1]

# Rearrange feature names so they match the sorted feature importances
names = [X.columns[i] for i in indices]

# Create plot
plt.figure()

# Create plot title
plt.title("Feature Importance")

# Add bars
plt.bar(range(X.shape[1]), importances[indices])

# Add feature names as x-axis labels
plt.xticks(range(X.shape[1]), names, rotation=90)

# Show plot
plt.show()
```



7 Conclusion

This paper presents a case study of Comptroller and Auditor General (CAG) of India to c heck the applicability of machine learning methods to predict the fraudulent firms during audit planning. A complete Audit Field Work Decision Support framework is proposed to help an auditor to decide the amount of field work required for a particular firm and to sk ip visiting low risk firms. Fraudulent firm prediction is an important step at the preliminar y stage of an audit planning as high-risk firms are targeted for the maximum audit investi gation during field engagement. After collecting the data of 777 firms from 14 different s ectors, it is cleaned, transformed, and useful risk factors are examined

So after creating all the different models and implementing them on training and testing d ata, the best model for prediction among all the models as of now is Random forest classi fier

For future works, we are targeting to improve the performance of the classifiers by the en semble machine learning approach (using a hybrid of the best performing classifiers). In t he next step, we offer the auditors to handle the last 10 years data of firms on the top of a dvance big data techniques like Hadoop, Spark, etc.

8 References

- Cosserat, G. 2009. Accepting the engagement and planning the audit. In Modern auditing, ed. G. Cosserat and N. Rodda, 3rd ed., 734–36. John Wiley & Sons.

- Tschakert, N. 2016. The next frontier in data analytics. Journal of Accountancy. Accessed September 12, 2016. http://www.journalofaccountancy.com/issues/2016/aug/data-analy tics-skills.html

- Tysiac, K. 2015. Data analytics helps auditors gain deep insight. Journal of Accountancy. Accessed September 12, 2016. http://www.journalofaccountancy.com/issues/2015/apr/dataanalytics-for-auditors.html.

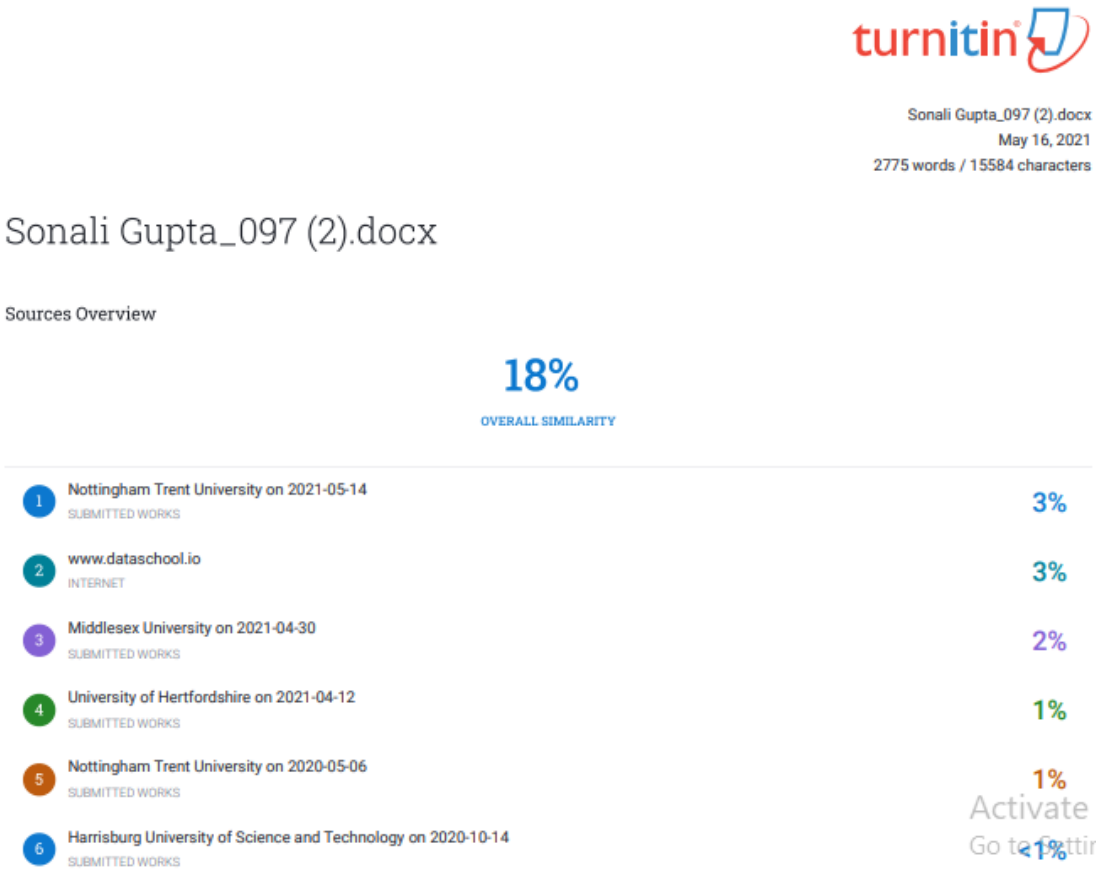
- Kotsiantis, S. 2006. Forecasting fraudulent financial statements using data mining. International Journal of Computational Intelligence 3 (2):104–10.

- Hooda, Nishtha, Seema Bawa, and Prashant Singh Rana. 'Fraudulent Firm Classification: A Case Study of an External Audit.' Applied Artificial Intelligence 32.1 (2018): 48-64.

- Fanning, K. M., and K. O. Cogger. 1998. Neural network detection of management fraud using published financial data. International Journal of Intelligent Systems in Accounting, Finance & Management 7 (1):21–41. doi:10.1002/(SICI)1099-1174(199803)7:13.0.CO;2-K

- Ali, S., and K. A. Smith. 2006. On learning algorithm selection for classification. Applied Soft Computing 6 (2):119–38. doi:10.1016/j.asoc.2004.12.002.

PLAGIARISM REPORT



turnitin

Sonali Gupta_097 (2).docx
May 16, 2021
2775 words / 15584 characters

Sonali Gupta_097 (2).docx

Sources Overview

18%
OVERALL SIMILARITY

1	Nottingham Trent University on 2021-05-14 SUBMITTED WORKS	3%
2	www.dataschool.io INTERNET	3%
3	Middlesex University on 2021-04-30 SUBMITTED WORKS	2%
4	University of Hertfordshire on 2021-04-12 SUBMITTED WORKS	1%
5	Nottingham Trent University on 2020-05-06 SUBMITTED WORKS	1%
6	Harrisburg University of Science and Technology on 2020-10-14 SUBMITTED WORKS	<1%