

**UJJAIN ENGINEERING COLLEGE**  
**UJJAIN (M.P.) -456010**



**Session 2019-20**

**Department of Mechanical Engineering**

**“TO PREDICT EMPLOYEE ATTRITION BASED ON  
PAST HISTORY”**

A major project submitted in partial fulfillment for award of degree of  
Bachelor of Engineering

*By*

**Anushree Anil (0701ME161010)**  
**Animesh Nema (0701ME161006)**  
**Kapil Vaishnav (0701ME161023)**  
**Adarsh Patidar (0701ME161003)**

*Under the Guidance of*

**Prof. LOKESH TIWARI**

## **CERTIFICATE**



This is to certify that the thesis entitled, “**TO PREDICT EMPLOYEE ATTRITION BASED ON PAST HISTORY**” submitted by **Anushree Anil, Animesh Nema, Kapil Vaishnav and Adarsh Patidar** in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Mechanical Engineering at Ujjain Engineering College, Ujjain is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge the matter embodied in the report has not been submitted to any other university/institution for the award of any degree.

**Dr. Manoj Gupta**  
HOD  
(Mechanical Engg. Dept.)

**Prof. Lokesh Tiwari**  
Assistant Professor &  
Project Guide  
(Mechanical Engg. Dept.)

## **ACKNOWLEDGEMENT**

I am indebted to my guide **Prof. Lokesh Tiwari** for giving me an opportunity to work under his guidance. Like a true mentor, he motivated and inspired me through the entire duration of my Work. I express my gratitude towards our dept. HOD **Prof. Manoj Gupta** sir for his support throughout the project work. I also extend my thanks to the supportive staff of Mechanical department for providing me all the necessary facilities to accomplish this project. Last but not the least, I express my profound gratitude to the Almighty and my parents for their blessings and support without which this task could have never been accomplished.

## **DECLARATION**

We, the undersigned solemnly declare that the project report of “**To Predict Employee Attrition Based on Past History**” is based on our own work carried out during the course of our study under the supervision of **Prof. Lokesh Tiwari**. We assert the statements made and conclusions drawn are an outcome of our research work. We further certify that the work contained in the report has been done by us under the general supervision of our supervisor. The work has not been submitted to any other institution for any other degree/diploma/certificate in this university or any other university of India or abroad. We have followed the guidelines provided by the university in writing the report. Whenever we have used materials (data, theoretical analysis and text) from other sources, we have given due credit to them by giving their details in the references.

**Anushree Anil**  
**Animesh Nema**  
**Kapil Vaishnav**  
**Adarsh Patidar**



## **INTRODUCTION**

As SARS-Covid19 or corona virus, which was declared a pandemic by March 2020, started to show its presence in India. In order to protect our country which had reported very few cases till then our Prime Minister announced a nationwide lockdown on 22th March with a short notice of 4 hrs which resulted in the world's stringent lockdown lasting more than 2 months with 1.3 billion peoples under house. The effect of lockdown become clear in the upcoming months In sundry ways out of which one is job losses. Because of this long unprecedented stand-still of economic activities employers started lay-off to keep themselves afloat in this tough times. There is huge loss of jobs in both formal and in formal sectors. As there is paucity of data for informal sector , we worked on Formal sector particularly IT sector.

## **OBJECTIVE**

As the COVID-19 keeps unleashing its havoc, the world continues to get pushed into the crisis of the great economic recession, more and more companies start to cut down their underperforming employees. Companies firing hundreds and thousands of Employees is a typical headline today. Cutting down employees or reducing an employee salary is a tough decision to take. It needs to be taken with utmost care as imprecision in the identification of employees whose performance is attriting may lead to sabotaging of both employees' career and the company's reputation in the market. Hence we built a Machine learning Model on the dataset provided by IBM to aid Human Resource Department.

## **MACHINE LEARNING**

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programmes that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computer to learn automatically without human intervention or assistance and adjust actions accordingly.

Machine learning is about extracting knowledge from data. It is a research field at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning. The application of machine learning methods has in recent years become ubiquitous in everyday life. From automatic recommendations of which movies to watch, to what food to order or which products to buy, to personalized online radio and recognizing your friends in your photos, many modern websites and devices have machine learning algorithms at their core. When you look at a complex website like Facebook, Amazon, or Netflix, it is very likely that every part of the site contains multiple machine learning models.

Machine learning is the idea that there are generic algorithms that can tell you something interesting about a set of data without you having to write any custom code specific to the problem. Instead of writing code, you feed data to the generic algorithm and it builds its own logic based on the data.

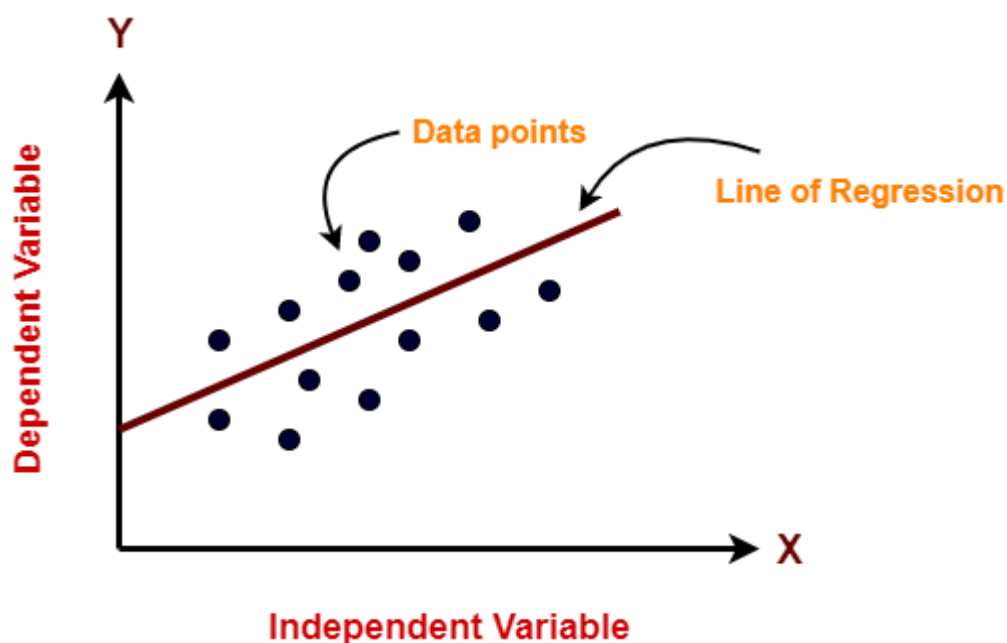
For example, one kind of algorithm is a classification algorithm. It can put data into different groups. The same classification algorithm used to recognize handwritten numbers could also be used to classify emails into spam and not-spam without changing a line of code. It's the same algorithm but it's fed different training data so it comes up with different classification logic.

“Machine learning” is an umbrella term covering lots of these kinds of generic algorithms. Some of these algorithms are described in next chapter which we have used in our model.

## VARIOUS ALGORITHMS AND PERFORMANCE MATRIX

1. **Linear Regression:** Linear regression is one of the very basic forms of machine learning where we train a model to predict the behaviour of your data based on some variables. In the case of linear regression as you can see the name suggests linear that means the two variables which are on the x-axis and y-axis should be linearly correlated.

Example: Make prediction about Cost of House based on area



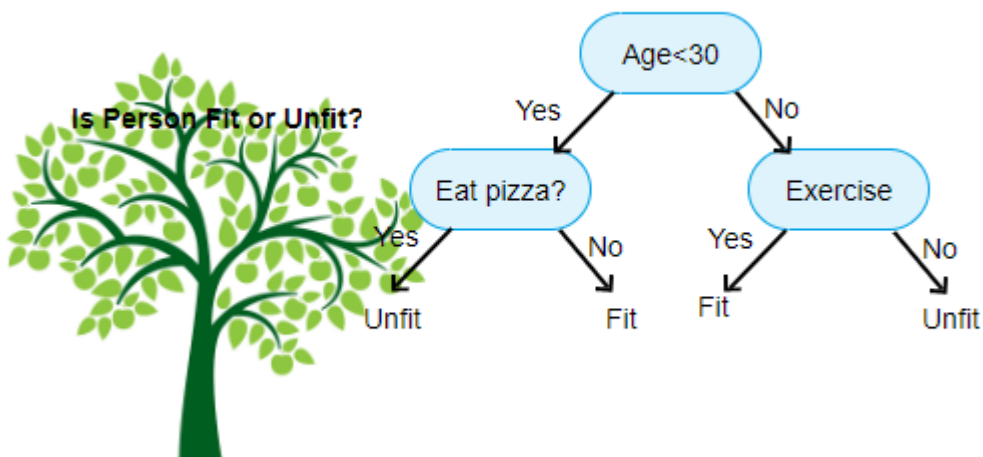
2. **Logistic Regression:** Logistic Regression, despite its name, is not an algorithm for regression problems but one of the widely used machine learning algorithms for binary or multinomial classification problems. It is the classification counterpart of linear regression. Logistic Regression is a predictive modelling algorithm for modelling binary categorical variables.

Example: Email classification as spam or not spam.





3. **Support Vector Machine:** Support Vector Machines, one of the most popular and widely used supervised machine learning algorithms. SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.
4. **Decision Tree Algorithm:** A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.



5. **Random Forest:** Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

Random forests has a variety of applications, such as recommendation engines, image classification and feature selection. It can be used to classify loyal loan applicants, identify fraudulent activity and predict diseases. It lies at the base of the Boruta algorithm, which selects important features in a dataset.

### Measure for Performance of Model

Confusion Matrix; Confusion Matrix is a performance measurement for machine learning classification. In the case of classification problem having only one classification accuracy might not give you the whole picture. So, a confusion matrix or error matrix is used for summarizing the performance of a classification algorithm.

A confusion matrix provides a summary of the predictive results in a classification problem. Correct and incorrect predictions are summarized in a table with their values and broken down by each class.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

### Definition of the Terms:

**True Positive:** You predicted positive and it’s true. You predicted that an animal is a cat and it actually is.

**True Negative:** You predicted negative and it's true. You predicted that animal is not a cat and it actually is not (it's a dog).

**False Positive (Type 1 Error):** You predicted positive and it's false. You predicted that animal is a cat but it actually is not (it's a dog).

**False Negative (Type 2 Error):** You predicted negative and it's false. You predicted that animal is not a cat but it actually is.

**Classification Accuracy:**

Classification Accuracy is given by the relation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

**Recall (Sensitivity):**

Recall is defined as the ratio of the total number of correctly classified positive classes divide by the total number of positive classes. Or, out of all the positive classes, how much we have predicted correctly. Recall should be high.

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{or} \quad \frac{\text{True Positive}}{\text{Actual Results}}$$

**Precision:**

Precision is defined as the ratio of the total number of correctly classified positive classes divided by the total number of predicted positive classes. Or, out of all the predictive positive classes, how much we predicted correctly. Precision should be high.

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{or} \quad \frac{\text{True Positive}}{\text{Predictive Results}}$$

*Trick to remember: Precision has Predictive Results in the denominator.*

**F-score or F1-score:**

It is difficult to compare two models with different Precision and Recall. So to make them comparable, we use F-Score. It is the Harmonic Mean of Precision and Recall. As compared to Arithmetic Mean, Harmonic Mean punishes the extreme values more. F-score should be high.

$$\mathbf{F - score = \frac{2 * Recall * Precision}{Recall + Precision}}$$

### **Specificity:**

Specificity determines the proportion of actual negatives that are correctly identified.

$$\mathbf{Specificity = \frac{TN}{TN + FP}}$$

### **AUC-ROC:**

In Machine Learning, performance measurement is an essential task. So when it comes to a classification problem, we can count on an AUC - ROC Curve. When we need to check or visualize the performance of the multi - class classification problem, we use AUC (**Area Under The Curve**) ROC (**Receiver Operating Characteristics**) curve. It is one of the most important evaluation metrics for checking any classification model's performance. It is also written as AUROC (**Area Under the Receiver Operating Characteristics**)

AUC - ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. By analogy, Higher the AUC, better the model is at distinguishing between patients with disease and no disease.

The ROC curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.



## **ABOUT DATA**

For training and testing our ML model we used data from IBM. Our training data was consists of 1628 entry on 29 parameter including attrition and our test data was of 470.

- Id - an anonymous id given to an Employee
- Age - Age of an Employee
- Attrition - Did the Employee leave the company, 0-No, 1-Yes
- Business Travel – Travelling frequency of an Employee
- Department - Work Department
- Distance From Home - Distance of office from home
- Education Field - Field of Education
- Employee Number - Number of Employees in the division of a given Employee
- Environment Satisfaction - Work Environment Satisfaction
- Gender - Gender of Employee
- Marital Status - Martial Status of an employee
- Monthly Income - Monthly Income of Employee in USD
- Number of Companies Worked - Number of Companies in which Employee has worked before joining this Company
- Over Time - Does The person work overtime
- Percent Salary Hike - Average annual salary hike in percentages
- Stock Option Level - Company stocks given to an Employee
- Total Working Years - Total working experience of an employee
- Training Times Last Year - No. of trainings an employee went through last year
- Years At Company - Number of years worked at this company
- Years In Current Role - Number of years in current role
- Years Since Last Promotion - Number of years since last promotion
- Years With Current Manager - Number of years with the current manager
- Education
  - 1 'Below College'
  - 2 'College'
  - 3 'Bachelor'
  - 4 'Master'
  - 5 'Doctor'
- Environment Satisfaction
  - 1 'Low'
  - 2 'Medium'
  - 3 'High'
  - 4 'Very High'
- Job Involvement
  - 1 'Low'
  - 2 'Medium'
  - 3 'High'
  - 4 'Very High'

- Job Satisfaction
  - 1 'Low'
  - 2 'Medium'
  - 3 'High'
  - 4 'Very High'
- PerformanceRating
  - 1 'Low'
  - 2 'Good'
  - 3 'Excellent'
  - 4 'Outstanding'
- Behaviour
  - 1 'Good'
  - 2 'Bad'
  - 3 'Not Rated'
- CommunicationSkill
  - 1 'Bad'
  - 2 'Average'
  - 3 'Good'
  - 4 'Better'
  - 5 'Best'
- StockOptionLevel
  - 0 'No stocks'
  - 1 'Less Stocks'
  - 2 'Moderate Stocks'
  - 3 'A lot of Stocks'

## CALCULATION

In [1]:

```
import pandas as pd
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict, StratifiedKFold

import matplotlib.pyplot as plt
```

In [2]:

```
df= pd.read_csv('E:/Course/train.csv',pd.set_option('display.max_columns', None))
dt = pd.read_csv('E:/Course/test.csv',pd.set_option('display.max_columns', None))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Parser
Warning: Falling back to the 'python' engine because the 'c' engine does not
support sep=None with delim_whitespace=False; you can avoid this warning by
specifying engine='python'.
"""Entry point for launching an IPython kernel.
```

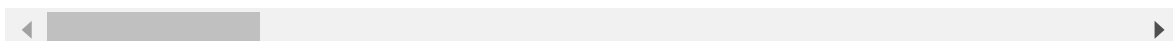
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel\_launcher.py:2: Parser  
Warning: Falling back to the 'python' engine because the 'c' engine does not  
support sep=None with delim\_whitespace=False; you can avoid this warning by  
specifying engine='python'.

In [3]:

df.head()

Out[3]:

	<b>ID</b>	<b>Age</b>	<b>Attrition</b>	<b>BusinessTravel</b>	<b>Department</b>	<b>DistanceFromHome</b>	<b>Education</b>	<b>Education</b>
<b>0</b>	1 & Development	30	0	Non-Travel	Research	<b>2</b>	3	M
<b>1</b>	2 & Development	36	0	Travel_Rarely	Research		12	4 Life Scie
<b>2</b>	3	55	1	Travel_Rarely	Sales		2	1 M
<b>3</b>	4 & Development	39	0	Travel_Rarely	Research		24	1 Life Scie
<b>4</b>	5 & Development	37	0	Travel_Rarely	Research		3	3





In

Out[4]:

In [5]:

```
print(df.shape)
print(dt.shape)
```

(1628, 29)

(470, 28)

In [6]:

```
from sklearn.preprocessing import LabelEncoder
```

In [7]:

```
le_BusinessTravel=LabelEncoder()
le_Department = LabelEncoder()
le_EducationField = LabelEncoder()
le_Gender = LabelEncoder()
le_MaritalStatus = LabelEncoder()
le_OverTime = LabelEncoder()
le_JobRole = LabelEncoder()
```

In [8]

```
df['BusinessTravel_n'] = le_BusinessTravel.fit_transform(df['BusinessTravel'])
df['Department_n'] = le_Department.fit_transform(df['Department'])
df['EducationField_n'] = le_EducationField.fit_transform(df['EducationField'])
df['Gender_n'] = le_Gender.fit_transform(df['Gender'])
df['MaritalStatus_n'] = le_MaritalStatus.fit_transform(df['MaritalStatus'])
df['OverTime_n'] = le_OverTime.fit_transform(df['OverTime'])
df['JobRole_n'] = le_JobRole.fit_transform(df['JobRole'])
```

In

In [10]:

```
df.drop(['MaritalStatus'],axis=1,inplace=True)
```

In [11]:

```
df.shape
```

Out[11]:

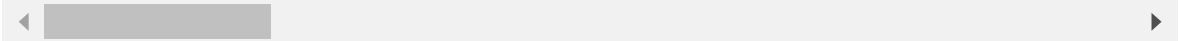
(1628, 27)

In [12]:

```
df.head()
```

Out[12]:

	Id	Age	Attrition	Education	EnvironmentSatisfaction	JobInvolvement	JobSatisfaction	M 0130	0	3
	3	3	4							
1	2	36	0	4		3	3	3		
2	3	55	1	1		3	3	4		
3	4	39	0	1		1	3	4		
4	5	37	0	3		3	3	3		



In [13]:

```
dt['BusinessTravel_n'] = le_BusinessTravel.fit_transform(dt['BusinessTravel'])
dt['Department_n'] = le_Department.fit_transform(dt['Department'])
dt['EducationField_n'] = le_EducationField.fit_transform(dt['EducationField'])
dt['Gender_n'] = le_Gender.fit_transform(dt['Gender'])
dt['MaritalStatus_n'] = le_MaritalStatus.fit_transform(dt['MaritalStatus'])
dt['OverTime_n'] = le_OverTime.fit_transform(dt['OverTime'])
dt['JobRole_n'] = le_JobRole.fit_transform(dt['JobRole'])
```

In [14]:

```
dt.drop(['MaritalStatus'],axis = 1,inplace=True)
```

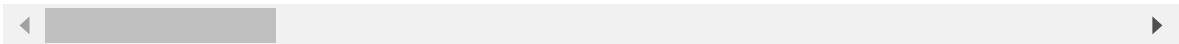
In [15]:

```
dt.drop(['BusinessTravel','Department','DistanceFromHome','EducationField','EmployeeNumber','Gender','JobRole','OverTime'],axis=1,inplace=True)
```

In [16]:

Out[16]:

	Id	Age	Education	EnvironmentSatisfaction	JobInvolvement	JobSatisfaction	MonthlyInc
0	1	28	3	4	3	4	2
1	2	31	4	1	4	4	5
2	3	37	3	3	4	1	5
3	4	42	2	4	2	4	6
4	5	45	2	3	3	2	4



In [17]:

```
dt.shape
```

Out[17]:

(470, 26)

In [18]:

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

In [19]:

```
X.shape
```

Out[19]:

(1628, 26)

In [20]:

```
df.shape
```

Out[20]:

(1628, 27)

In [21]:

```
print(X.shape)
```

(1628, 26)

(1628,)

In [22]:

```
from sklearn import
decomposition
pca =
decomposition.PCA()
pca.n_components = 2
pca_data = pca.fit_transform(X)
print('Shape of Reduced data=',pca_data.shape)
```

Shape of Reduced data= (1628, 2)

In [23]:

```
from sklearn.svm import SVC
```

In [24]:

```
clf = SVC(kernel='linear')
```

In [25]:

```
cv = StratifiedKFold(n_splits=10, shuffle = False, random_state = 76)
```

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model\_selection\\_split.py:296: FutureWarning: Setting a random\_state has no effect since shuffle is False. This will raise an error in 0.24. You should leave random\_state to its default (None), or set shuffle=True.  
FutureWarning

In [26]:

```
clf_logreg = LogisticRegression(max_iter=10000)
```

In [27]:

```
clf_logreg.fit(X, y)
```

Out[27]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=10000,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=
0,
warm_start=False)
```

In [28]:

```
y_pred_class_logreg = cross_val_predict(clf_logreg, X, y, cv = cv)
y_pred_prob_logreg = cross_val_predict(clf_logreg, X, y, cv = cv, method="predict_proba")
y_pred_prob_logreg_class1 = y_pred_prob_logreg[:, 1]
```

In [

In [29]:

```
clf_SGD = SGDClassifier(max_iter=10000)
```

In [30]:

```
clf_SGD.fit(X, y)
```

Out[30]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,  
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,  
  
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',  
              max_iter=10000, n_iter_no_change=5, n_jobs=None, penalty='l2',  
  
              power_t=0.5, random_state=None, shuffle=True, tol=0.001,  
              validation_fraction=0.1, verbose=0, warm_start=False)
```

In [31]:

```
y_pred_class_SGD = cross_val_predict(clf_SGD, X, y, cv = cv)  
y_pred_prob_SGD = cross_val_predict(clf_SGD, X, y, cv = cv, method="decision_function")
```

In [32]:

```
clf_rfc = RandomForestClassifier()
```

In [33]:

```
clf_rfc.fit(X, y)
```

Out[33]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,  
                       criterion='gini', max_depth=None, max_features='auto',  
  
                       max_leaf_nodes=None, max_samples=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=None, verbose=0,  
                       warm_start=False)
```

In [34]:

```
y_pred_class_rfc = cross_val_predict(clf_rfc, X, y, cv = cv)
```

In [35]:

```
y_pred_prob_rfc = cross_val_predict(clf_rfc, X, y, cv = cv, method="predict_proba")
y_pred_prob_rfc_class1 = y_pred_prob_rfc[:, 1]
```

In [36]:

```
from sklearn.base import BaseEstimator
import numpy as np

class BaseClassifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)

base_clf = BaseClassifier()
cross_val_score(base_clf, X, y, cv=10, scoring="accuracy").mean()
```

Out[36]:

0.5171779141104295

In [37]:

```
acc_logreg = cross_val_score(clf_logreg, X, y, cv = cv, scoring = 'accuracy').mean()
acc_SGD = cross_val_score(clf_SGD, X, y, cv = cv, scoring = 'accuracy').mean()
acc_rfc = cross_val_score(clf_rfc, X, y, cv = cv, scoring = 'accuracy').mean()

acc_logreg, acc_SGD, acc_rfc
```

Out[37]:

(0.865379080512005, 0.7512042717564189, 0.8487843671892753)

In [38]:

```
logloss_logreg = cross_val_score(clf_logreg, X, y, cv = cv, scoring = 'neg_log_loss').mean()
logloss_rfc = cross_val_score(clf_rfc, X, y, cv = cv, scoring = 'neg_log_loss').mean()
```

In [38]:

```
from sklearn.calibration import CalibratedClassifierCV

new_clf_SGD = CalibratedClassifierCV(clf_SGD)
new_clf_SGD.fit(X, y)
logloss_SGD = cross_val_score(new_clf_SGD, X, y, cv = cv, scoring = 'neg_log_loss').mean()

logloss_logreg, logloss_SGD, logloss_rfc
```

Out[39]:

(-0.5167302694383367, -0.6925157422810481, -0.29788469848989463)

In [40]:

```
fpr_logreg, tpr_logreg, thresholds_logreg = metrics.roc_curve(y, y_pred_prob_logreg_class1)
fpr_rfc, tpr_rfc, thresholds_rfc = metrics.roc_curve(y, y_pred_prob_rfc_class1)
fpr_SGD, tpr_SGD, thresholds_SGD = metrics.roc_curve(y, y_pred_prob_SGD)

plt.plot(fpr_logreg, tpr_logreg, label="logreg")
plt.plot(fpr_rfc, tpr_rfc, label="rfc")
plt.plot(fpr_SGD, tpr_SGD,
label="SGD") plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.rcParams['font.size'] = 12
plt.title('ROC curve')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.legend(loc="lower right",
fontsize=10) plt.grid(True)
```

In [41]:

```
def evaluate_threshold(tpr, fpr, clf_threshold, threshold):
    print('Sensitivity:', tpr[clf_threshold > threshold][-1])
    print('Specificity:', 1 - fpr[clf_threshold > threshold][-1])
```

In [42]:

```
Sensitivity: 0.889171974522293
Specificity: 0.6631079478054567
Sensitivity: 0.7133757961783439
Specificity: 0.970344009489917
```

Out[42]:

(None, None)

In [43]:

```
evaluate_threshold(tpr_SGD, fpr_SGD, thresholds_SGD, 0.2), evaluate_threshold(tpr_SGD,
fpr_SGD, thresholds_SGD, 0.8)
```

```
Sensitivity: 0.6127388535031847
Specificity: 0.8279952550415184
Sensitivity: 0.6127388535031847
Specificity: 0.8279952550415184
```

Out[43]:

(None, None)

In [44]:

```
roc_auc_logreg = cross_val_score(clf_logreg, X, y, cv = cv, scoring = 'roc_auc').mean()
roc_auc_SGD = cross_val_score(clf_SGD, X, y, cv = cv, scoring = 'roc_auc').mean()
roc_auc_rfc = cross_val_score(clf_rfc, X, y, cv = cv, scoring = 'roc_auc').mean()

roc_auc_logreg, roc_auc_SGD, roc_auc_rfc
```

Out[44]:

(0.9852504711719158, 0.9033574897014958, 0.9800452488687782)

In [45]:

```
from sklearn.ensemble import RandomForestClassifier m_rfc
= RandomForestClassifier()
m_rfc.fit(X,y)
```

Out[45]:

RandomForestClassifier(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,
criterion='gini', max\_depth=None, max\_features='auto',

```
max_leaf_nodes=None, max_samples=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=None,
verbose=0, warm_start=False)
```



```
m_rfc.score(X,y)
```

Out[46]:

In [47]:

```
pred_clf_logreg = clf_logreg.predict(dt)
pred_clf_SGD = clf_SGD.predict(dt)
pred_clf_rfc = clf_rfc.predict(dt)
```

In [50]:

pred\_clf\_rfc

Out[50]:

[illegible]

In [51]:

```
out=(clf_rfc.predict_proba(dt))[:,1]
```

In [54]:

```
import numpy
```

In [55]:

```
fin = numpy.asarray([out])
numpy.savetxt("fin.csv",fin,delimiter=",")
```

Out[56]

```
array([0.06, 0.12, 0.1 , 0.25, 0.07, 0.2 , 0.48, 0.27, 0.11, 0.06, 0.11,
       0.27, 0.13, 0.02, 0.06, 0.08, 0.41, 0.13, 0.08, 0.02, 0.08, 0.49,
       0.15, 0.11, 0.06, 0.01, 0.05, 0.03, 0.76, 0.1 , 0.07, 0.18, 0.32,
       0.23, 0.03, 0.1 , 0.11, 0.27, 0.22, 0.26, 0.63, 0.25, 0.09, 0.09,
       0.07, 0.11, 0.12, 0.21, 0.01, 0.14, 0.46, 0.08, 0.17, 0.31, 0.29,
       0.16, 0.09, 0.5 , 0.01, 0.13, 0.07, 0.08, 0.37, 0.01, 0.1 , 0.15,
       0.07, 0.11, 0.09, 0.15, 0.17, 0.03, 0.08, 0.16, 0.24, 0.34, 0.06,
       0.11, 0.03, 0.42, 0.06, 0.18, 0.08, 0.02, 0.15, 0.12, 0.16, 0.21,
       0.16, 0.32, 0.19, 0.12, 0.27, 0.09, 0.6 , 0.19, 0.25, 0.07, 0.12,
       0. , 0.1 , 0.07, 0.16, 0.13, 0.15, 0.08, 0.06, 0.13, 0.1 , 0.1 ,
       0.23, 0.21, 0.09, 0.12, 0.16, 0.02, 0.07, 0.04, 0.05, 0.09, 0.22,
       0.23, 0.04, 0.21, 0.25, 0.06, 0.03, 0.28, 0.55, 0.17, 0.2 , 0.05,
       0.15, 0.12, 0.1 , 0.08, 0.69, 0.08, 0.31, 0.04, 0.16, 0.2 , 0.05,
       0.02, 0.07, 0.37, 0.14, 0.53, 0.17, 0.27, 0.33, 0.16, 0.12, 0.52,
       0.12, 0.08, 0.03, 0.1 , 0.11, 0.07, 0.46, 0.14, 0.08, 0.06, 0.11,
       0.25, 0.36, 0.19, 0.02, 0.09, 0.05, 0.15, 0.06, 0.18, 0.02, 0.17,
       0.14, 0.17, 0.2 , 0.46, 0.21, 0.16, 0.61, 0.15, 0.42, 0.17, 0.13,
       0.03, 0.12, 0.13, 0.14, 0.08, 0.11, 0.48, 0.24, 0.31, 0.08, 0.17,
       0.07, 0.13, 0.42, 0.24, 0.4 , 0.05, 0.14, 0.03, 0.42, 0.21, 0.08,
       0.27, 0.38, 0.15, 0.13, 0.09, 0.27, 0.75, 0.48, 0.08, 0.11, 0.08,
       0.05, 0.13, 0.15, 0.29, 0.25, 0.04, 0.12, 0.1 , 0.06, 0.1 , 0.34,
       0.17, 0.07, 0.42, 0.07, 0.33, 0.12, 0.09, 0.45, 0.04, 0.13, 0.04,
       0.33, 0.42, 0.12, 0.03, 0.51, 0.05, 0.08, 0.17, 0.03, 0.04, 0.05,
       0.12, 0.05, 0.44, 0.04, 0.17, 0.1 , 0.31, 0.05, 0.04, 0.27, 0.31,
       0.28, 0.26, 0.05, 0.02, 0.36, 0.09, 0.07, 0.1 , 0.59, 0.31, 0.39,
       0.23, 0.08, 0.23, 0.09, 0.16, 0.23, 0.21, 0.16, 0.13, 0.13, 0.49,
       0.1 , 0.16, 0.03, 0.11, 0.22, 0.06, 0.05, 0.09, 0.09, 0.12, 0.08,
       0.15, 0.51, 0.12, 0.03, 0.04, 0.09, 0.53, 0.07, 0.08, 0.27, 0.07,
       0.17, 0.12, 0.09, 0.21, 0.19, 0.08, 0.18, 0.14, 0.42, 0.14, 0.05,
       0.21, 0.2 , 0.1 , 0.05, 0.2 , 0.56, 0.1 , 0.35, 0.38, 0.29, 0.43,
       0.04, 0.23, 0.08, 0.07, 0.23, 0.13, 0.38, 0.12, 0.11, 0.42, 0.09,
       0.49, 0.07, 0.19, 0.18, 0.08, 0.09, 0.04, 0.03, 0.1 , 0.02, 0.01,
       0.2 , 0.13, 0.06, 0.43, 0.19, 0.08, 0.06, 0.45, 0.01, 0.14, 0.08,
       0.08, 0.46, 0.13, 0.07, 0.12, 0.16, 0.35, 0.1 , 0.05, 0.08, 0.52,
       0.2 , 0.16, 0.23, 0.31, 0.38, 0.01, 0.55, 0.08, 0.04, 0.07, 0.42,
       0.27, 0.42, 0.49, 0.22, 0.45, 0.06, 0.01, 0.2 , 0.1 , 0.06, 0.16,
       0.39, 0.04, 0.04, 0.08, 0.09, 0.05, 0.21, 0.2 , 0.37, 0.1 , 0.13,
       0.11, 0.11, 0.17, 0.02, 0.04, 0.06, 0.16, 0.14, 0.01, 0.61, 0.21,
       0.09, 0.07, 0.28, 0.23, 0.05, 0.09, 0.06, 0.07, 0.14, 0.06, 0.17,
       0.02, 0.04, 0.01, 0.1 , 0.1 , 0.13, 0.01, 0.04, 0.02, 0.09, 0.03,
       0.06, 0.11, 0.09, 0.01, 0.11, 0.02, 0.05, 0.19, 0.04, 0.16, 0.11,
       0.07, 0.66, 0.16, 0.13, 0.01, 0.4 , 0.28, 0.4 , 0.11, 0. , 0.07,
       0.2 , 0.23, 0.42, 0.36, 0.77, 0.3 , 0.06, 0.1 ])
```

In [57]:

```
import numpy as np
a = np.asarray([ out ])
a.tofile('fish.csv',sep=',',format='%10.5f')
```

In [58]:

```
#%10.5f
```

## RESULT

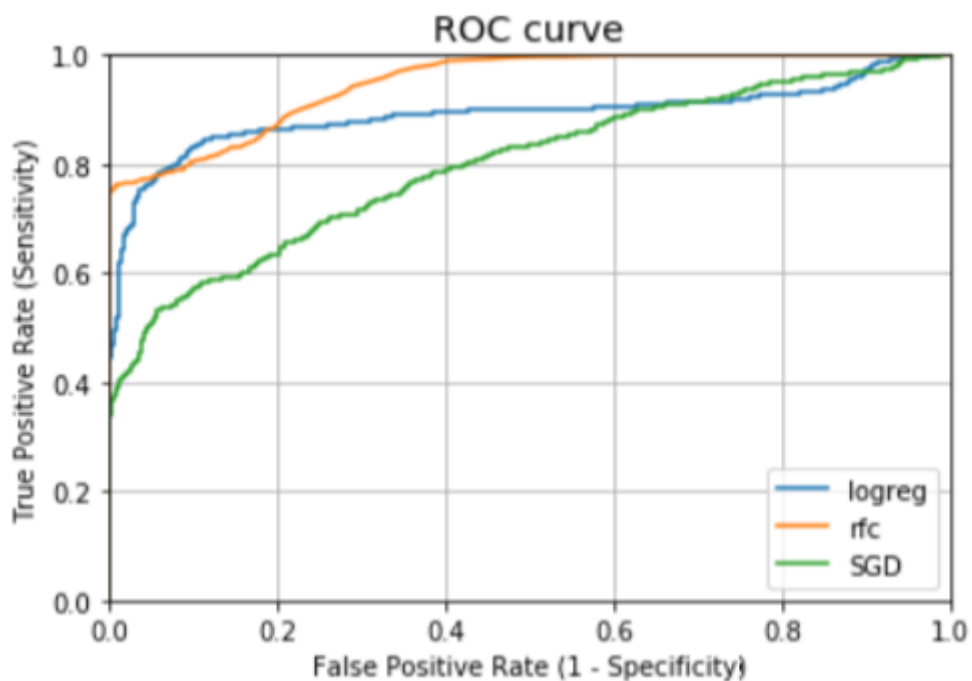
We built three different model on the same set of data having 28 parameters, using Logistic Regression, SGD Classifier (SVM) and Random Forest Classifier and got Following results on Confusion Matrix.

Logistic Regression: 98.52%

SGD Classifier: 90.33%

Random Forest Classifier: 98.00%

As all our model are neither under fitting nor over fitting and forming well on train set data. We recommend to use Random Forest Classifier over Logistic Regression because of its Flexibility.



## **CONCLUSION**

It is a paradox that even after achieving 98% accuracy we cannot rely completely on a Machine Learning Model for such a critical decision for both employee and employer. Still it can assist human Resource department to make decision based on past performance of employee. And In future we can refine this model further by adding more data.

