1. EXECUTIVE SUMMARY

Putting simply, employee attrition is the reduction of staff by voluntary or involuntary reasons. These can be through natural means like retirement, or it can be through resignation, termination of contract, or when a company decides to make a position redundant.

Data analytics is useful for prediction from more information and it helps HR department to predict whether a particular employee will quit or not. Huge amount of employee related data is maintained on monthly basis. The stored data can be useful source of predicting the occurrence of future employee attrition. Some of the data mining and machine learning techniques are used to predict the attrition, such as multiple regression, decision tree, random forest etc. This paper provides an insight of the existing algorithm and it gives an overall summary of the existing work.

2. RESEARCH OBJECTIVE

This work aims in developing a Decision Support System in employee attrition detection that uses the data mining technique having best accuracy and performance among Support Vector Machine, KNN, and Random Forest etc. By using several HR employee data management system parameters such as satisfaction level, last evaluation, number of projects, average monthly hours and salary it is possible to measure the possibility of quitting by an employee. For deriving the algorithm with the best accuracy in the detection and prediction of employee attrition, a comparative analysis of chosen machine learning algorithms has been shown. This algorithm takes the parameters such as satisfaction level, last evaluation, number of projects, average monthly hours, salary, departments etc. as input and shows the probability of an employee leaving an organization as an output. And finally, deploying the best model API.

3. Technique Used

Random Forest: Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

Decision Tree : decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

KNN: KNN (K — Nearest Neighbors) is one of many (supervised learning) algorithms used in data mining and machine learning, it’s a classifier algorithm where the learning is based “how similar” is a data (a vector) from other.

SVM: In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

4. RESEARCH PLAN AND METHODOLOGY

In this study, an efficient machine learning algorithm was chosen from some available algorithms in, ‘python is an interpreted, high-level, general-purpose programming language. It was created by Guido van Rossum and first released in 1991, Python’s design philosophy emphasizes code readability with its notable use of significant whitespace. In order to detect the presence or to decide the probability of an employee quitting form a large dataset. The step by step workflow of the complete system have been mentioned below:

- Collection and selection of different employee attrition datasets.
- Cleaning the dataset to be used, train various machine learning algorithm.
- Comparison of various data mining algorithm’s accuracy and performance in predicting employee quitting his/ her job.
- Finally selecting the best algorithm based on the performance characteristics of the models.

5. ANALYSIS OF DATA AND FINDINGS

The dataset taken for data mining application includes 14 kinds of input which are as follows

NO.	ATTRIBUTE	DESCRIPTION
1	Satisfaction Level	Employee is satisfied or not with his or her work (values range from 0 to 1). 0 stands for not satisfied and 1 stand for satisfied.
2	Last Evaluation	Last rating of employee value in the columns (values range from 0 to 1).
3	Number of Projects	Number of project employee is working on (2, 3, 4, 5, 6, 7 projects) .
4	Average Monthly Hours	Average monthly hour spent
5	Time spent at the Company	Number of years spent in the company (2, 3, 4, 5, 6, 7, 8, 10 years).
6	Whether they have had a work accident	During working did employee have an accident 0 - no and 1 for yes.
7	Whether they have had a promotion in the last 5 years	promotion in last 5 years 0 - no and 1 for yes
8	Department	Employee working in which departments (sales, accounting, technical, support, IT, product_mng, marketing, HR, R&D, management).

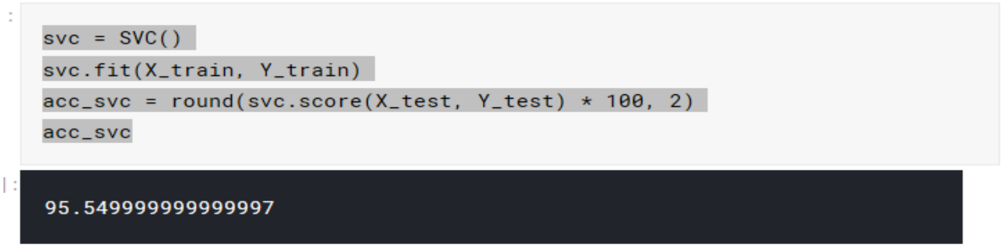
9	Salary	Salary of the employees is categorized in three levels (low medium and high).
10	Left	Employee has left the company or not 0 - no and 1 for yes.

Table:1 Blue Print of the dataset

There are various kinds of algorithms are available, which can be applied to the dataset. But in this report, I have only applied 4 machine learning algorithm and for the performance measure I have used the following criteria:

For SVM and KNN I have only used Score as the criteria since the main file has no mention of these

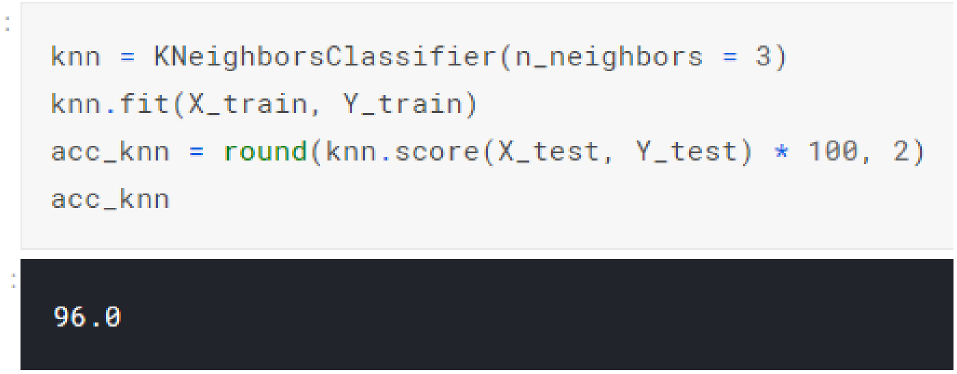
SVM SCORE:



```
svc = SVC()
svc.fit(X_train, Y_train)
acc_svc = round(svc.score(X_test, Y_test) * 100, 2)
acc_svc
```

```
95.549999999999997
```

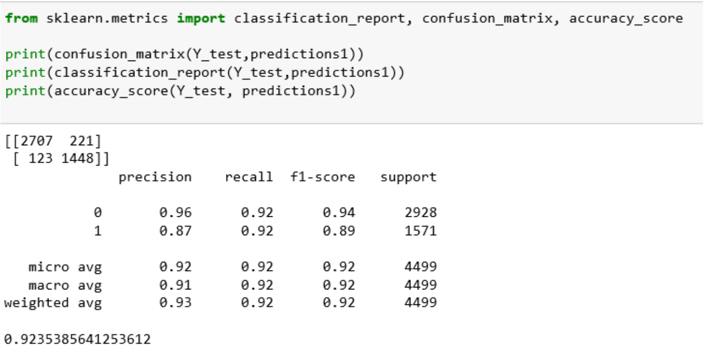
KNN



```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
acc_knn = round(knn.score(X_test, Y_test) * 100, 2)
acc_knn
```

```
96.0
```

Decision Tree

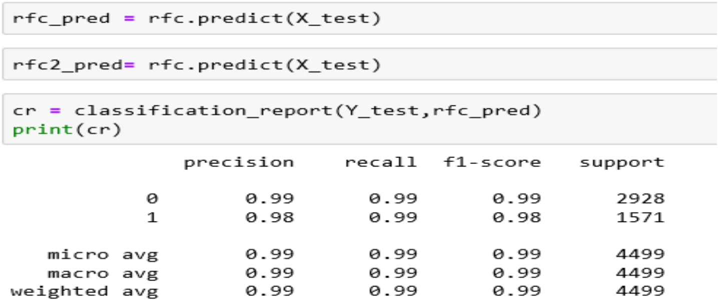


```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(Y_test,predictions1))
print(classification_report(Y_test,predictions1))
print(accuracy_score(Y_test, predictions1))
```

```
[[2707  221]
 [ 123 1448]]
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      2928
           1       0.87      0.92      0.89      1571

   micro avg       0.92      0.92      0.92      4499
   macro avg       0.91      0.92      0.92      4499
weighted avg       0.93      0.92      0.92      4499

0.9235385641253612
```

Random Forest

Accuracy: 0.989



```
rfc_pred = rfc.predict(X_test)
```

```
rfc2_pred= rfc.predict(X_test)
```

```
cr = classification_report(Y_test,rfc_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2928
           1       0.98      0.99      0.98      1571

   micro avg       0.99      0.99      0.99      4499
   macro avg       0.99      0.99      0.99      4499
weighted avg       0.99      0.99      0.99      4499
```

5. CONCLUSION

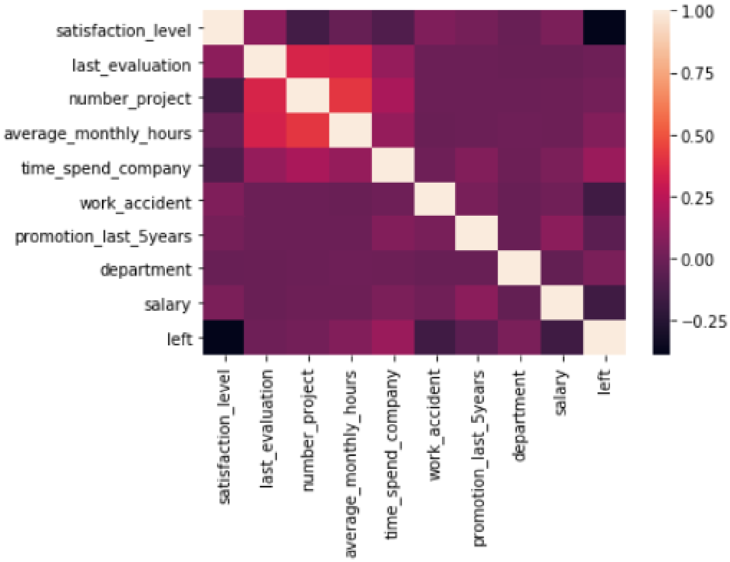
Through my analysis, managing the level of satisfaction is the key to keep employees with the firm. This is especially important for employees who have been around for more than 3 years. Other than that the employee evaluation and number of projects should also be monitored. This firm’s HR Head would do well to craft programs to keep tabs and these metrics so as to have a successful talent retention policy.

6. RECOMMENDATIONS

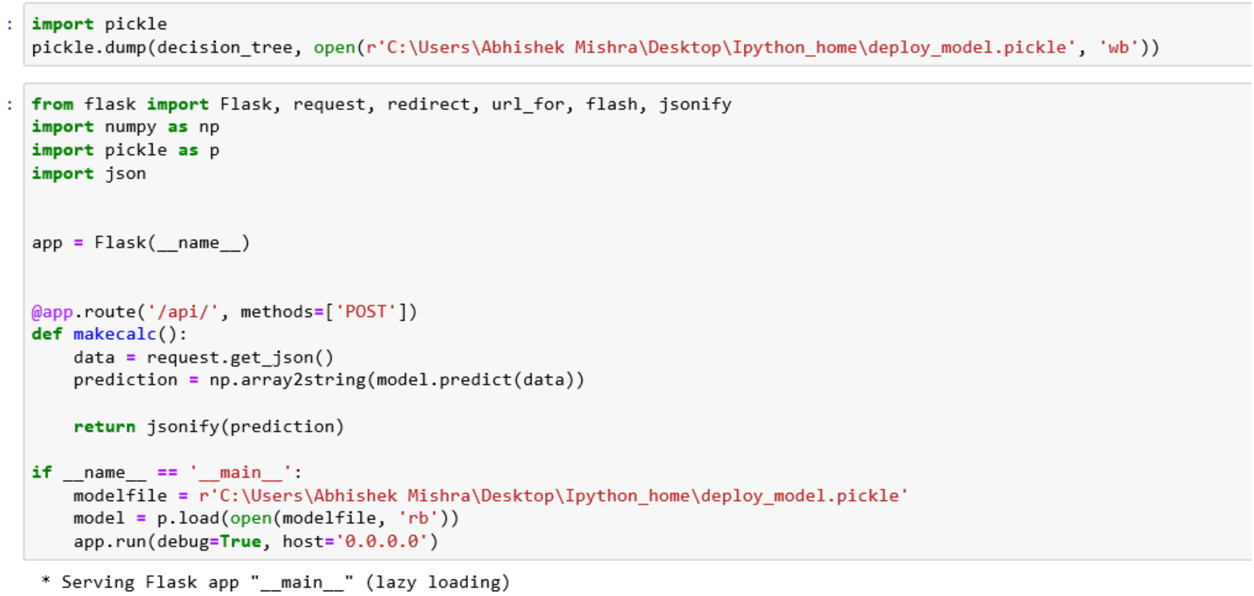
At, then end I would like the firm to adopt my ML API, as it will not only allow them to save cost due untimely quitting of job by an employee. But also able to retain them. And further this can be share with others companies or subsidiaries and prove to be commercial success.

7. Appendix

Heat Map



Code for applying the ML model



```
: import pickle
  pickle.dump(decision_tree, open(r'C:\Users\Abhishek Mishra\Desktop\Ipython_home\deploy_model.pickle', 'wb'))
```

```
: from flask import Flask, request, redirect, url_for, flash, jsonify
  import numpy as np
  import pickle as p
  import json


  app = Flask(__name__)


  @app.route('/api/', methods=['POST'])
  def makecalc():
      data = request.get_json()
      prediction = np.array2string(model.predict(data))

      return jsonify(prediction)

  if __name__ == '__main__':
      modelfile = r'C:\Users\Abhishek Mishra\Desktop\Ipython_home\deploy_model.pickle'
      model = p.load(open(modelfile, 'rb'))
      app.run(debug=True, host='0.0.0.0')

   * Serving Flask app "__main__" (lazy loading)
```