Predicting Employee Turnover

By Satyam Tandon

A. Epilogue

No company likes loosing their valuable employees. It is an arduous and expensive task to replace a long term employee for any company. According to Employee Benefit News(EBN), on average it costs employers about 33% of the employee's annual salary or about $15,000 for a worker earning a median salary of $45,000 to replace them. This does not include non monetary costs such as time spent finding and training a replacement.

Moreover , according to Work Institute's 2017 Retention Report , a study of 34,000 respondent's concluded that 75% of causes of turnover are preventable. More information on employee turnover can be found in this article by HR Drive.

B. Objective

In order to reduce employee turnover, first we need to identify the employees that are at risk of turnover, for a company. This project uses universal metrics that are available to almost any organization such as an employee's satisfaction levels, score on last evaluation, average monthly hours, years at company, work accidents, their current project number for the company,wether they have been promoted in the last 5 years , their department and salary levels. None of these features about their employees is very hard to obtain for a company of any size.

C. The Data

The dataset used for this project is freely available to use on kaggle.com. It is a relatively clean dataset that contains 15,000 rows, each representing an employee and 9 features about each employee mentioned above in a CSV file that can be imported into a pandas data frame.

	satisfaction_level	last_evaluation	project_number	average_monthly_hours	years_at_company	work_accidents	attrition	promotion_last_5years	department	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low

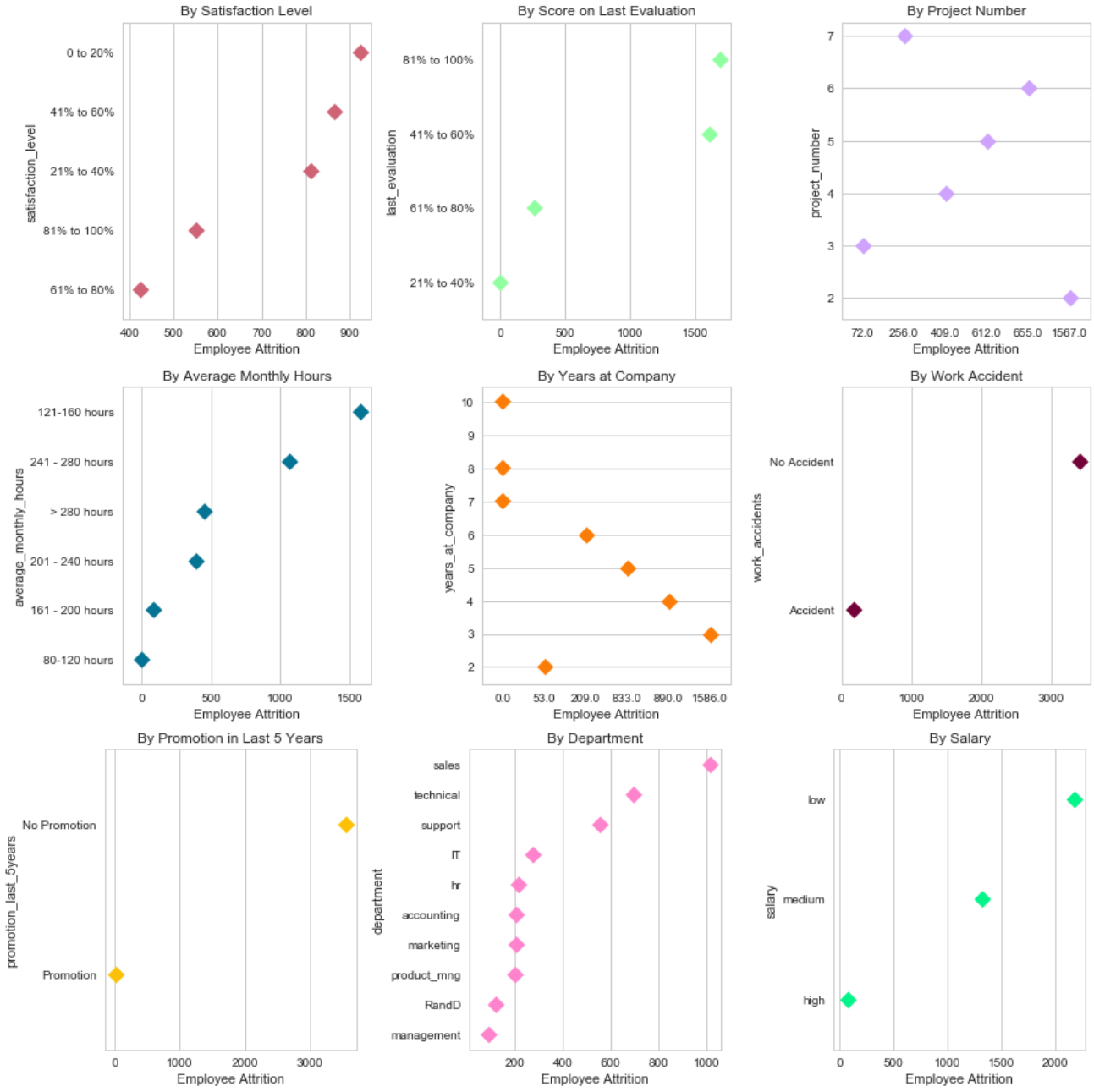
Since the dataset is quiet clean, I simply imported it into a pandas data frame and renamed some of the columns for easier interpretability. However the data will require some pre-processing before machine learning algorithms can be fit on it.

The dataset contains 11,428 instances of the employee staying and 3,571 instances of the employee leaving , which is approximately a **76% - 24%** split for

the binary target variable, which is workable with most robust machine learning ensemble methods that will be used for this project.

E. Target Variable Breakdown by Features

To breakdown total employee turnover by each feature variable I wrote a function that creates a new grouped data frame , converting unique values to categorical variables in each feature column where necessary, using ranges for continuous variable and aggregates the grouped data frame by the count of the target variable, i.e employee attrition and plotted them on a seaborn stripplot.



Key Findings:

A. Surprising mostly employees who got a score above 81% on their last evaluation have the highest rate of turnover among all ranges of evaluation scores, which could indicate most employees leave for better or new opportunities rather than being unhappy with the company.
B. Among all the departments, sales has the highest turnover with lowest turnover for management positions.
C. As expected, employees with higher salaries have the lowest turnover among all salary brackets while the low salaried employees have a higher turnover. Employees who were promoted in the last 5 years have a lower turnover than employees who were not promoted in the last 5 years.
D. Employees who work on average 121-160 hours a month or approximately 24 - 32 hours a week have the highest turnover while employees who work on an average 80 - 120 hours a month or approximately 16-24 hours a week have the lowest turnover
E. Employees who have worked for 10 years at the same company have the lowest turnover which could indicate they are in management positions which as we saw has the lowest turnover rate while employees in their third year have the highest rate of turnover

F. Pre-Processing the Data for Machine Learning Algorithms

Binary categorical features such as Promotion and Work Accident have already been correctly label encoded including the target variable, attrition. I further **label encoded the feature Salary**, replacing low, medium and high with 1,2 and 3 respectively. However the **categorical feature , department cannot be label encoded as it would not produce the best results , therefore I will use one hot encoding** to covert each unique value in the department feature variable into it's own column with a binary value of 1 or 0, and dropping one new column entirely to avoid double counting for robust results.

One Hot Encoding and Label Encoding appropriate columns

In [5]:
```
employee_data.loc[:,'salary'].replace(['high','medium','low'],[3,2,1],inplace=True)
employee_data_p = pd.get_dummies(employee_data, drop_first=True)
employee_data_p.head(5)
```

Out[5]:

salary	department_RandD	department_accounting	department_hr	department_management	department_marketing	department_product_mng	department_sales
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1

G. Ensemble Methods

After aforementioned pre-processing, the data is ready for machine learning algorithms. I split the dataset into **80% training set and 20% testing/hold out set** to test how well the algorithms have generalized after I cross validated using 5 fold cross validation and tuned the appropriate hyper parameters for each algorithm. I will be using robust ensemble methods such as voting, bagging and boosting to train my models. I will use the scikit learn python library.

G.1 Voting Classifier

Models:

- Support Vector Classifier(Tuned Parameters: C and Gamma)
- K-Nearest Neighbor (Tuned Parameter: n_neighbors)
- Decision Tree Classifier (Tuned Parameter: min_samples_split and min_samples_leaf)
- Logistic Regression

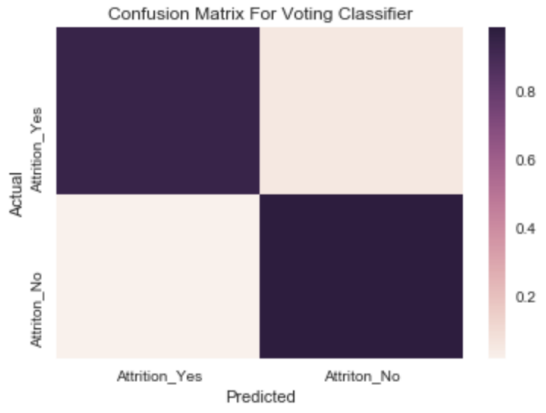
Voting: Softmax

Results on hold out set:

```
Training Accuracy: 97.7499500173 %

-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

Accuracy on unseen hold out set: 97.5666666667 %
```




```
Binary confusion matrix:
Predicted      Attrition_Yes  Attriton_No  __all__
Actual
Attrition_Yes            668           38      706
Attriton_No               35         2259     2294
__all__                  703         2297     3000
```

```
Positive = Attriton_No(employee did not leave)
Negative = Attriton_Yes(employee left)

True Positive Rate: 98.4742807323 %
False Positive Rate: 5.38243626062 %
True Negative Rate: 94.6175637394 %
False Negative Rate: 1.52571926765 %
```

Key Observation and Interpretation:
Overall the model has generalized very well, and most likely will do so mostly for other ensemble methods as well. **The metric of importance that would be beneficial to improve on would be the False Positive Rate , which is when the model classifies an employee staying when they actually left**. The **False Positive Rate for the voting classifier is 5.38%** or 38 misclassified observations.The True Negative Rate is quiet robust but it would be in the best interest of the objectives of this project to **lower the False Positive Rate as much as possible without sacrificing other metrics too much or overfilling the model.**

G.2 Bagging

The next ensemble method I used was bagging or bootstrap aggregating that aggregates the prediction of several weak models trained on random subsets of data with replacement.

G.2.1 Random Forest Classifier

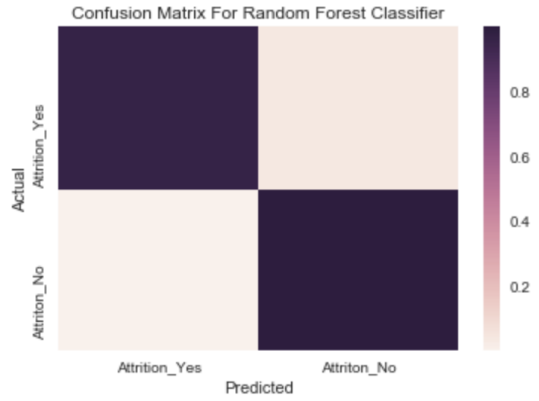
Hyper-parameters tuned:
- min_samples_leaf
- min_samples_split
- max_features
- max_depth

After tuning with 20 estimators, I increased the number of estimators to 200 and used the algorithm to make predictions on the hold out set

```
Training Accuracy: 99.7333111093 %

-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

Accuracy on unseen hold out set: 98.8333333333 %
```




```
Binary confusion matrix:
Predicted      Attrition_Yes  Attriton_No  __all__
Actual
Attrition_Yes            678           28      706
Attriton_No                7         2287     2294
__all__                  685         2315     3000

Positive = Attriton_No(employee did not leave)
Negative = Attriton_Yes(employee left)
```

```
True Positive Rate: 99.6948561465 %
False Positive Rate: 3.96600566572 %
True Negative Rate: 96.0339943343 %
False Negative Rate: 0.305143853531 %
```

Key Observation:
The random forest model performed better than the voting classifier in terms of the False Positive Rate. The True Negative Rate also increased to 96.03%. The model misclassified 28 employees as staying who left, bringing down the **False Positive Rate from 5.38% to 3.96%**

G.2.2 Extra Trees Classifier

The next bagging algorithm I used was the extra trees classifier which is similar to the random forest classifier except it splits the nodes of each individual tree on random features.
Hyper-parameters Tuned:

- min_samples_leaf
- min_samples_split
- max_depth
- max_features

I tuned the extra trees classifier on 20 estimators and increased the number to 200 before fitting on the hold out set.

```
Training Accuracy: 99.9916659722 %

-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

Accuracy on unseen hold out set: 98.6666666667 %
```


```
Binary confusion matrix:
Predicted      Attrition_Yes  Attriton_No  __all__
Actual
Attrition_Yes            676           30      706
Attriton_No               10         2284     2294
__all__                  686         2314     3000

Positive = Attriton_No(employee did not leave)
Negative = Attriton_Yes(employee left)

True Positive Rate: 99.5640802092 %
False Positive Rate: 4.2492917847 %
True Negative Rate: 95.7507082153 %
False Negative Rate: 0.435919790759 %
```

Key Observation:
The extra trees classifier performed better on the training set but did **slightly worse than the random forest classifier on the test set and in terms of the False Positive Rate at 4.24%.**

G.3 Boosting

The next ensemble method I used on the dataset is boosting, where again several weak models are fit on the dataset wherein each new model attempts to improve upon the inaccuracies of the previous model.

G.3.1 AdaBoost

Hyper-parameters Tuned:
- Individual decision tree parameters: min_samples_split, min_samples_leaf
- Boosting parameters: n_learners, learning rate

```
Mean Training Accuracy: 96.4913311803 %

-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

Accuracy on unseen hold out set: 96.6666666667 %
```


```
Binary confusion matrix:
Predicted      Attrition_Yes  Attriton_No  __all__
Actual
Attrition_Yes            666           40      706
Attriton_No               60         2234     2294
__all__                  726         2274     3000

Positive = Attriton_No(employee did not leave)
Negative = Attriton_Yes(employee left)

True Positive Rate: 97.3844812554 %
False Positive Rate: 5.6657223796 %
True Negative Rate: 94.3342776204 %
False Negative Rate: 2.61551874455 %
```

Key Observations:
The model performed worse than the bagging methods, with the **highest False Positive Rate out of all the models so far at 5.66%**

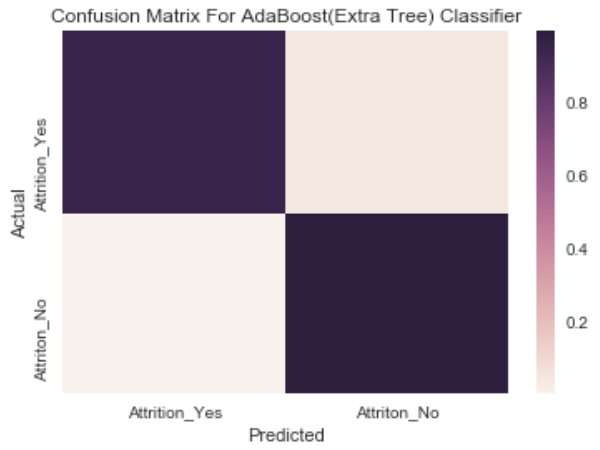
G.3.2 AdaBoost with Extra Tree Classifier

Using Extra tree classifier instead of decision trees as each individual model for AdaBoost.
Hyper-parameters Tuned:
- Individual decision tree parameters: min_samples_split, min_samples_leaf
- Boosting parameters: n_learners, learning rate

```
Mean Training Accuracy: 98.8166381597 %

-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

Accuracy on unseen hold out set: 98.5 %
```




```
Binary confusion matrix:
Predicted      Attrition_Yes  Attriton_No  __all__
Actual
Attrition_Yes            674           32      706
Attriton_No               13         2281     2294
__all__                  687         2313     3000

Positive = Attriton_No(employee did not leave)
Negative = Attriton_Yes(employee left)

True Positive Rate: 99.433304272 %
False Positive Rate: 4.53257790368 %
True Negative Rate: 95.4674220963 %
False Negative Rate: 0.566695727986 %
```

Key Observation:
The model performed slightly **better than Adaboost with decision trees**, lowering the **False Positive Rate at 4.53%** but still has lower accuracy than previous models.

G.3.3 Stochastic Gradient Boosting

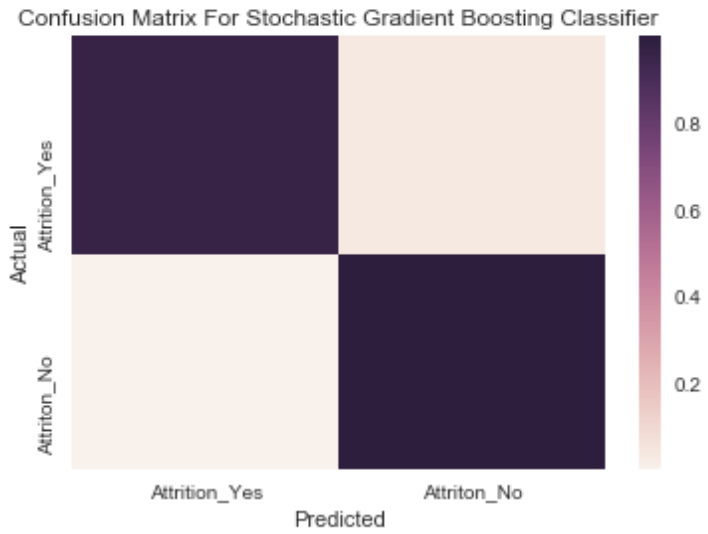
Stochastic gradient boosting also uses decision trees, but each individual tree is only fitted using a selected subsample of the dataset and a subsample of the total features instead of all the features.

Hyper-parameters Tuned:
- Individual decision tree parameters: min_samples_split, min_samples_leaf,max_depth
- Boosting parameters: n_learners, learning rate,subsample,max_features

```
Mean Training Accuracy: 99.0499994213 %

-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

Accuracy on unseen hold out set: 98.8333333333 %
```



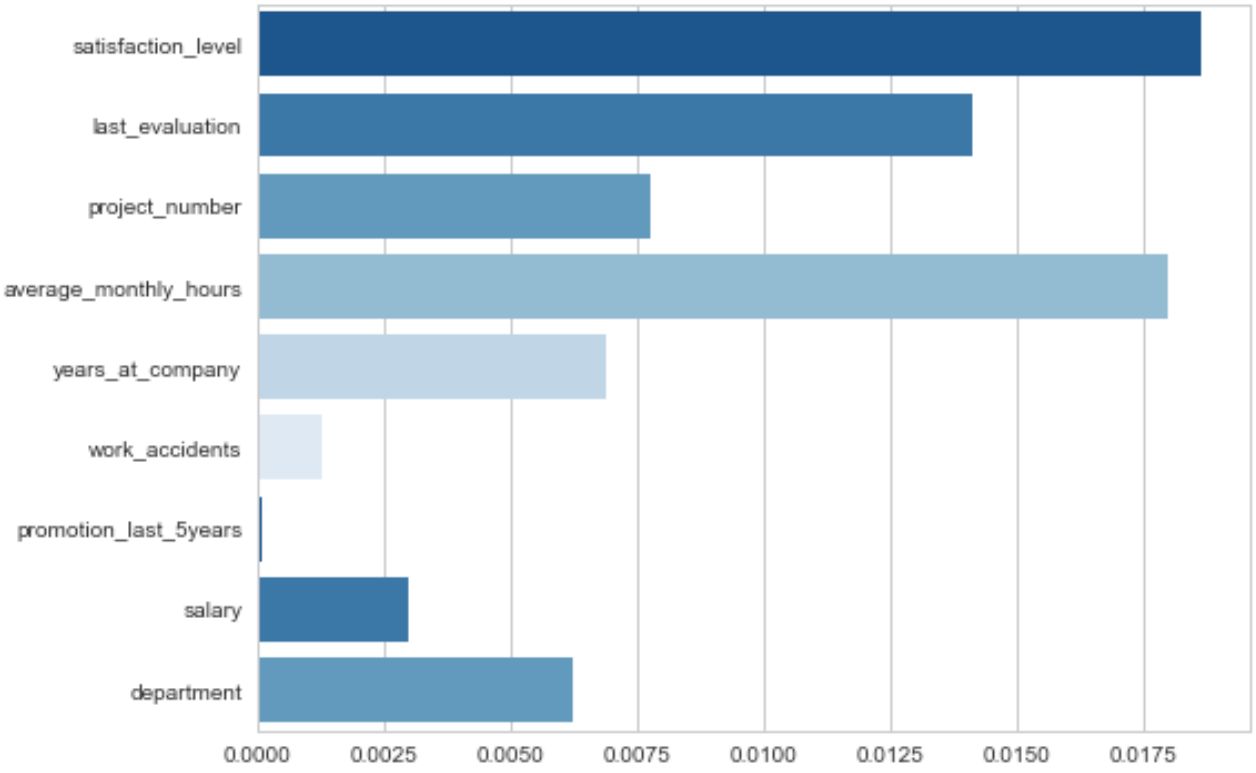

```
Binary confusion matrix:
Predicted      Attrition_Yes  Attriton_No  __all__
Actual
Attrition_Yes            681           25      706
Attriton_No               10         2284     2294
__all__                  691         2309     3000

Positive = Attriton_No(employee did not leave)
Negative = Attriton_Yes(employee left)

True Positive Rate: 99.5640802092 %
False Positive Rate: 3.54107648725 %
True Negative Rate: 96.4589235127 %
False Negative Rate: 0.435919790759 %
```

Key Observations:
The stochastic gradient boosting method has **performed better than all the other models in terms of overall accuracy and lowest False Positive Rate of 3.54%.**

G.4 Feature Importance



Key Observation:
The **top 3 most important features** as identified by the stochastic gradient boosting model, which is the best performing model are **satisfaction level, average monthly hours and score on last evaluation, respectively.**

H. Recommendations

I. Small Size Organizations:

This model was trained on 15,000 observations, for which small size organizations would require a lot of historical data , going back several years. For effective results , at least 5,000 observations to train the model should be used. Small size organizations should use the latest data, leading up to a few years to the current year for testing and tuning their models and making changes as necessary. If possible to acquire, data from organizations in similar industries who maintain similar metrics about their employees can also be useful.

II. Medium and Large Size Organizations:

It would be easier for medium and large size organizations to acquire enough data from recent years to train and tune models. Latest data should be continuously integrated into the model to maintain/improve model accuracy for best results. Changes in feature importance in making predictions should also be continuously tracked which could help identify important trends within the organization.

III. Additional Features:

This project used 9 features, but collecting and maintaining additional information for any size organization about employees such as age, sex, distance from home, educational background, education level, relationship satisfaction, business travel, stock option level, work life balance, years in current role, years with current manager, just to name a few and integrating them into predictive models could help greatly improve model accuracy and tracking the importance of these features could ultimately help shape company policy, culture and develop effective ways to provide important incentives to employees to reduce turnover and retain valuable employees.