INFO6105 Final Project

Team 4

1. Nidhi Singh
2. Meghana Yadav Gopalakrishna
3. Pragnya Kondrakunta

Employee Attrition - ¶

The cost of replacing an employee is quite large and we want to use data science to strategize around employee rentention. We picked an employee dataset that captures various factors of their working conditions. We have a column called `Attrition` that recorded whether an employee quit or not. Using this data, we want to identify the crucial factors that make an employee quit.

Hypothesis

People who are satisfied with their job are less likely to leave the company.

Additionally, we want to identify key drivers of attrition. We want to analyse the relation of other parameters like number of years worked at this company, monthly pay, the employee's age to attrition.

Contents

2. Exploratory Data Analysis
    - Correlation Matrix
    - Data Encoding
3. Model Training & Testing
    - Predict whether a person will leave (Probability of leaving)
4. Compare various models and their accuracy
    - Logistic Regression
    - Random Forest
    - SVMs
    - Artifical Neural Networks
5. Feature selection — Correlation and P-value
    - Figuring out the most significant parameters that contribution to a person leaving
6. Hypothesis Testing
7. Conclusion

1. Importing Libraries & Data Cleaning

In [1]:
```
#importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_absolute_error
```

In [2]:
```
#Load the data
data = pd.read_csv('./data/employees.csv')
# data.columns

#Dropping the columns with constant data
data.drop(['EmployeeCount', 'StandardHours', 'Over18', 'EmployeeNumber'] , axis=1

data.head()
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 31 columns

In [3]:
```python
#Get the number of rows and columns in the data
data.shape
```

Out[3]: (1470, 31)

> There are 1470 rows and 31 columns of employee data. Of these, 16 are numerical and 15 are categorical data

In [4]:
```python
def checkMissingData(DataFrame):
    if DataFrame.isna().sum().any() or DataFrame.isnull().values.any():
        print("Missing data exists")
    else:
        print("No missing data")

checkMissingData(data)
```

```
No missing data
```

> We have identified that there is no missing data (NA or null values). So we can proceed with EDA ...

2. Exploratory Data Analysis

Let's take a preliminary look at our dataset. Below we are calculating some statistical values for all columns

In [5]:
```
data.describe()
```

Out[5]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	Hourl
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.0(
mean	36.923810	802.485714	9.192517	2.912925	2.721769	65.8!
std	9.135373	403.509100	8.106864	1.024165	1.093082	20.3:
min	18.000000	102.000000	1.000000	1.000000	1.000000	30.0(
25%	30.000000	465.000000	2.000000	2.000000	2.000000	48.0(
50%	36.000000	802.000000	7.000000	3.000000	3.000000	66.0(
75%	43.000000	1157.000000	14.000000	4.000000	4.000000	83.7!
max	60.000000	1499.000000	29.000000	5.000000	4.000000	100.0(

8 rows × 23 columns

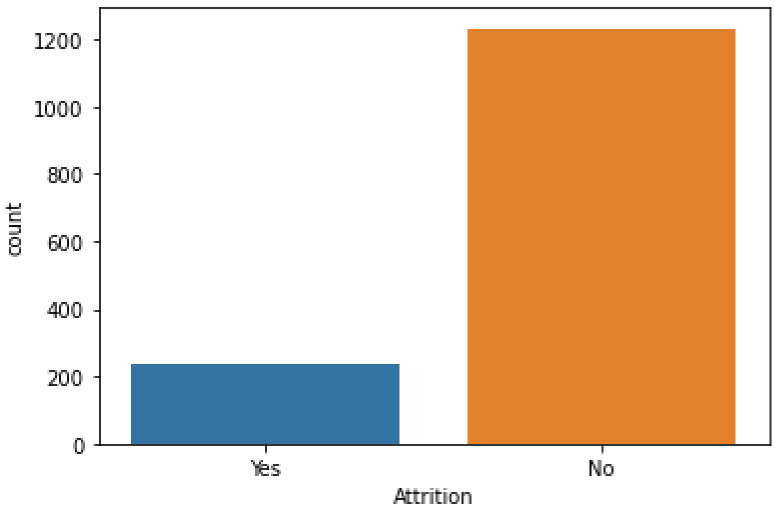
Basic statistical details such as percentile, mean, standard deviation for all the columns individually.

In [6]:
```
#Count of number of employee attrition; the number of employees stayed(no) and th
data['Attrition'].value_counts()
```

Out[6]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

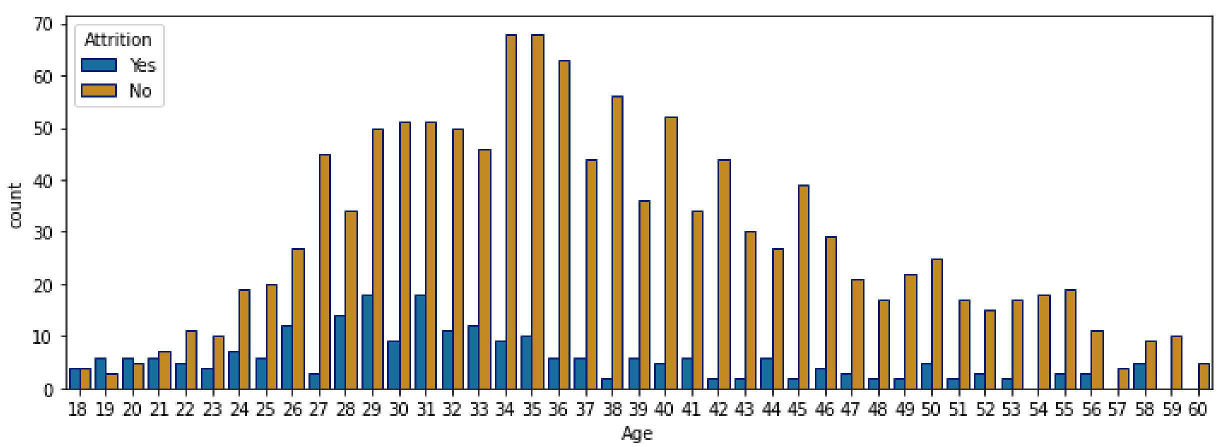
In [7]:
```
#Visualise the employee attrition count
sns.countplot(x = data['Attrition'])
```

Out[7]: <AxesSubplot:xlabel='Attrition', ylabel='count'>



In [8]:
```
#Number employees that left and stayed by age
fig_dims = (12,4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.countplot(x ='Age', hue='Attrition', data = data, palette = 'colorblind' , ax
```

Out[8]: <AxesSubplot:xlabel='Age', ylabel='count'>

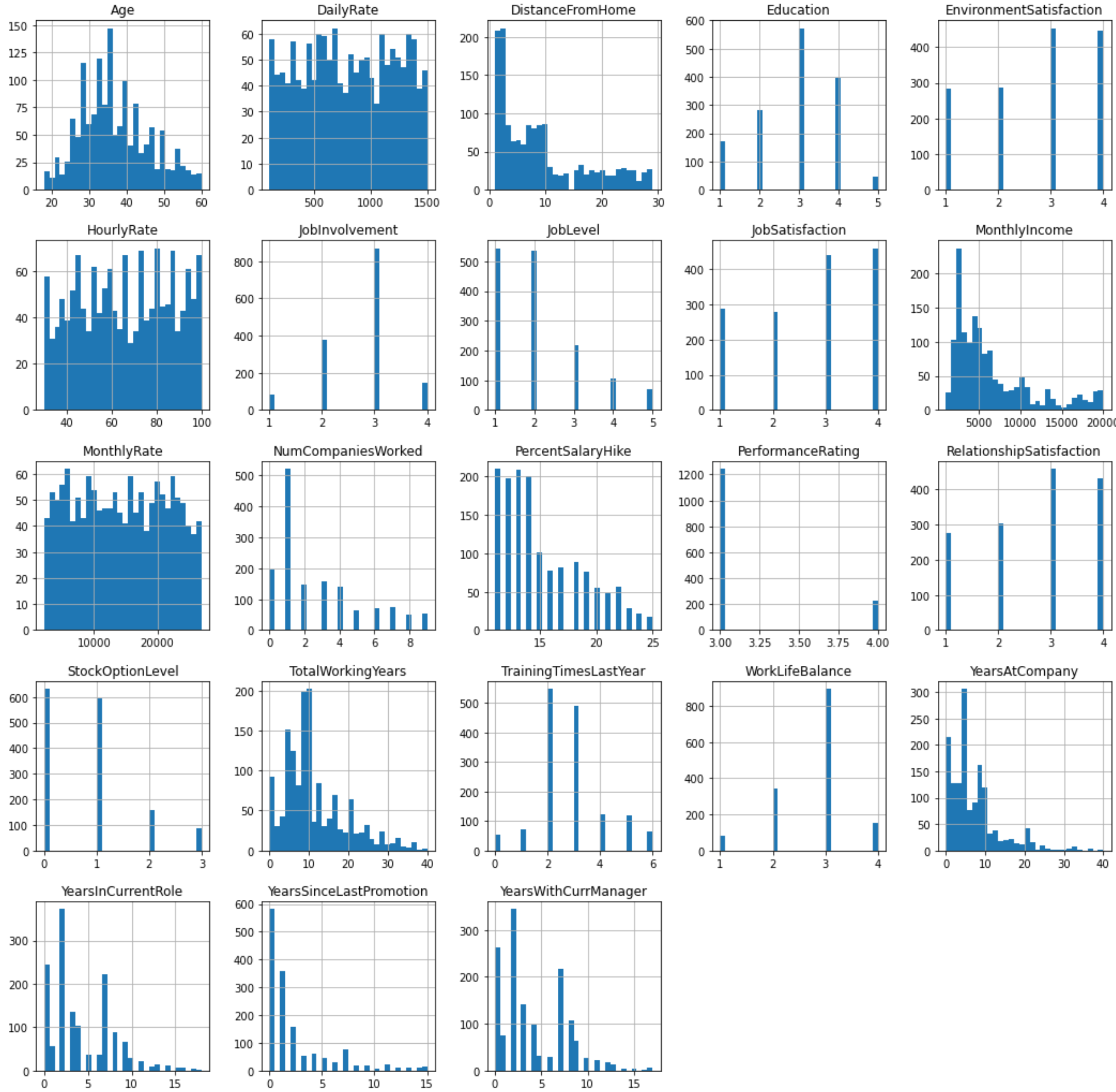


From the above graph the highest count of employee attrition is age 29 & 31. The age with the highest retention is age 34 & 35.

In [9]:
```python
#Shows distribution of the data
fig=plt.figure(figsize=(18,18))
ax=fig.gca()
data.hist(ax=ax,bins=30)
```

```
C:\Users\Pragnya\AppData\Local\Temp/ipykernel_35616/3375493487.py:4: UserWarnin
g: To output multiple subplots, the figure containing the passed axes is being
cleared
  data.hist(ax=ax,bins=30)
```

Out[9]:
```
array([[<AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'DailyRate'}>,
        <AxesSubplot:title={'center':'DistanceFromHome'}>,
        <AxesSubplot:title={'center':'Education'}>,
        <AxesSubplot:title={'center':'EnvironmentSatisfaction'}>],
       [<AxesSubplot:title={'center':'HourlyRate'}>,
        <AxesSubplot:title={'center':'JobInvolvement'}>,
        <AxesSubplot:title={'center':'JobLevel'}>,
        <AxesSubplot:title={'center':'JobSatisfaction'}>,
        <AxesSubplot:title={'center':'MonthlyIncome'}>],
       [<AxesSubplot:title={'center':'MonthlyRate'}>,
        <AxesSubplot:title={'center':'NumCompaniesWorked'}>,
        <AxesSubplot:title={'center':'PercentSalaryHike'}>,
        <AxesSubplot:title={'center':'PerformanceRating'}>,
        <AxesSubplot:title={'center':'RelationshipSatisfaction'}>],
       [<AxesSubplot:title={'center':'StockOptionLevel'}>,
        <AxesSubplot:title={'center':'TotalWorkingYears'}>,
        <AxesSubplot:title={'center':'TrainingTimesLastYear'}>,
        <AxesSubplot:title={'center':'WorkLifeBalance'}>,
        <AxesSubplot:title={'center':'YearsAtCompany'}>],
       [<AxesSubplot:title={'center':'YearsInCurrentRole'}>,
        <AxesSubplot:title={'center':'YearsSinceLastPromotion'}>,
        <AxesSubplot:title={'center':'YearsWithCurrManager'}>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



In [10]:
```python
print('Size of Full dataset is: {}'.format(data.shape))
```

```
Size of Full dataset is: (1470, 31)
```

2.1 Data Encoding (One Hot Encoding)

Since our dataset has quite a lot of categorical data, we need to encode strings(/categories) to numerical values that the machine can understand. We first check the columns that need encoding and subsequently, we use one hot encoding to encode those columns.

In [11]:
```
need_encoding_columns = [i for i in data.columns if type(data[i][0]) == str]
need_encoding_columns
```

Out[11]:
```
['Attrition',
 'BusinessTravel',
 'Department',
 'EducationField',
 'Gender',
 'JobRole',
 'MaritalStatus',
 'OverTime']
```

In [12]:
```
def one_hot_encode_cols(columns, dataframe):
    result_df = pd.DataFrame(dataframe)
    for i in columns:
        encoded_cols = pd.get_dummies(dataframe[i], drop_first = True)
        result_df.drop(columns=i, axis = 1,inplace=True)
        result_df = result_df.join(encoded_cols)

        if(encoded_cols.columns.all() == 'Yes'): result_df.rename(columns={'Yes':
    return result_df
```

In [13]:
```
oh_en_df = one_hot_encode_cols(need_encoding_columns, data)

# oh_en_df['Attrition'].index
```

2.2 Correlation Matrix

We want to capture the correlation of various parameters with each other.

In [14]:
```
#Correlation of the matrix
correlation_matrix = data.corr()
correlation_matrix
```

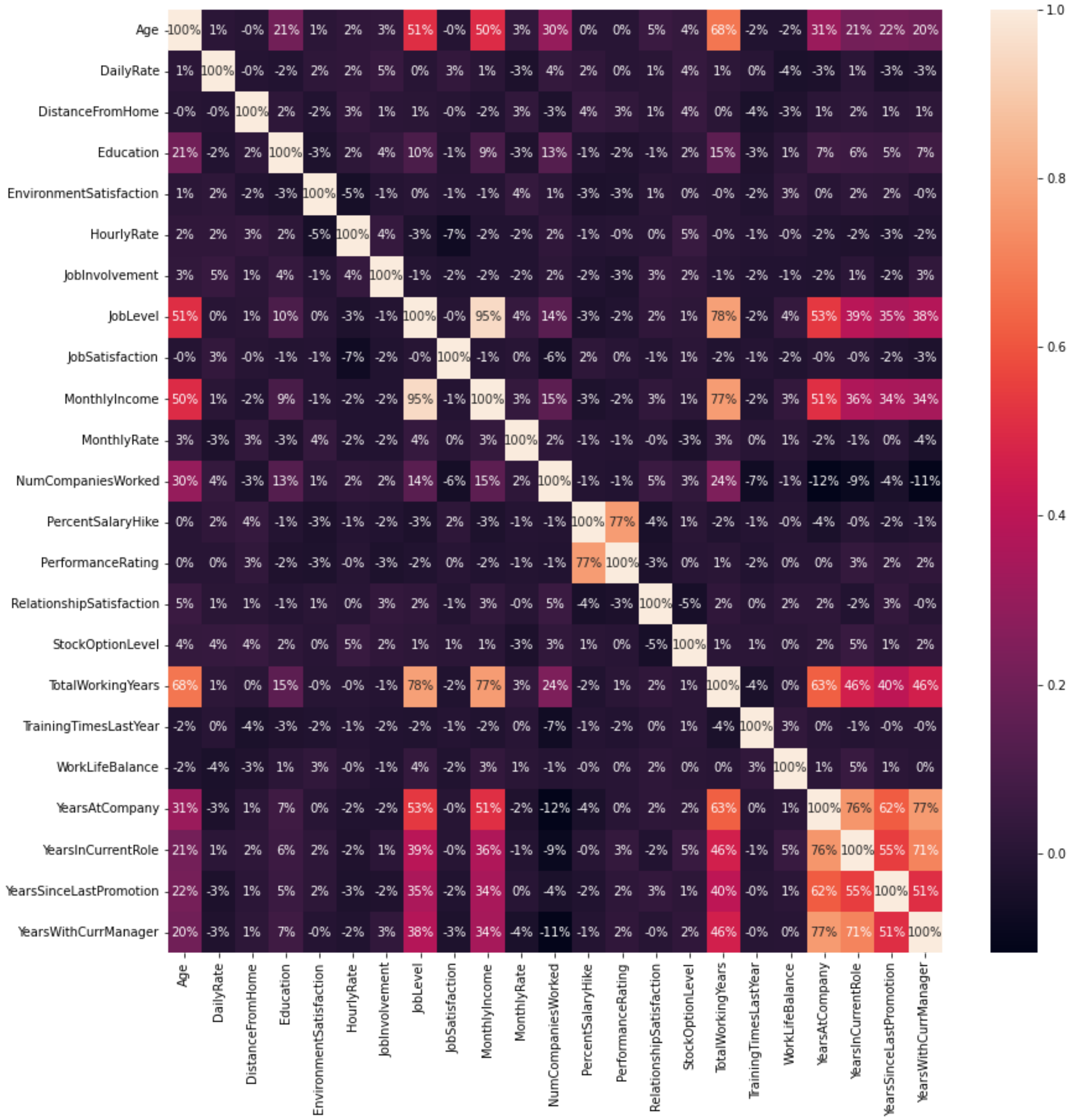
Out[14]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfact
Age	1.000000	0.010661	-0.001686	0.208034	0.010
DailyRate	0.010661	1.000000	-0.004985	-0.016806	0.018
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	-0.016
Education	0.208034	-0.016806	0.021042	1.000000	-0.027
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	1.000
HourlyRate	0.024287	0.023381	0.031131	0.016775	-0.049
JobInvolvement	0.029820	0.046135	0.008783	0.042438	-0.008
JobLevel	0.509604	0.002966	0.005303	0.101589	0.001
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	-0.006
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	-0.006
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	0.037
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	0.012
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	-0.031
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	-0.029
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	0.007
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	0.003
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	-0.002
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	-0.019
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	0.027
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	0.001
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	0.018
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	0.016
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	-0.004

23 rows × 23 columns

In [15]:
```
#Visualisation of the correlation
plt.figure(figsize = (14,14))
sns.heatmap(correlation_matrix,annot = True, fmt = '.0%')
```

Out[15]: <AxesSubplot:>



3. Model Training & Testing

In [16]:
```python
# Independent variables
X = oh_en_df.drop('Attrition', axis = "columns")

# Target/Dependent variable
y = oh_en_df['Attrition']
```

In [17]:
```python
X
```

Out[17]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvo
0	41	1102	1	2	2	94	
1	49	279	8	1	3	61	
2	37	1373	2	2	4	92	
3	33	1392	3	4	4	56	
4	27	591	2	1	1	40	
...	...	...	...	...	...	...	
1465	36	884	23	2	3	41	
1466	39	613	6	1	4	42	
1467	27	155	4	3	2	87	
1468	49	1023	2	3	4	63	
1469	34	628	8	3	2	82	

1470 rows × 44 columns

In [18]:
```python
y
```

Out[18]:
```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition, Length: 1470, dtype: uint8
```

In [19]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state =0 , test_
```

In [20]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1102, 44)
(368, 44)
(1102,)
(368,)
```

3. 1. Logistic Regression

In [21]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=0)
```

In [22]:
```
a =logreg.fit(X_train, y_train)
predicted_y = logreg.predict(X_test)
```

```
C:\Users\Pragnya\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:
763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

In [23]:
```
len(predicted_y)
```

Out[23]:
```
368
```

In [24]:
```
logreg.score(X_train,y_train)
print('Accuracy: {:.2f}%'.format(logreg.score(X_test, y_test)*100))
ac3 = logreg.score(X_train,y_train)
```

```
Accuracy: 84.51%
```

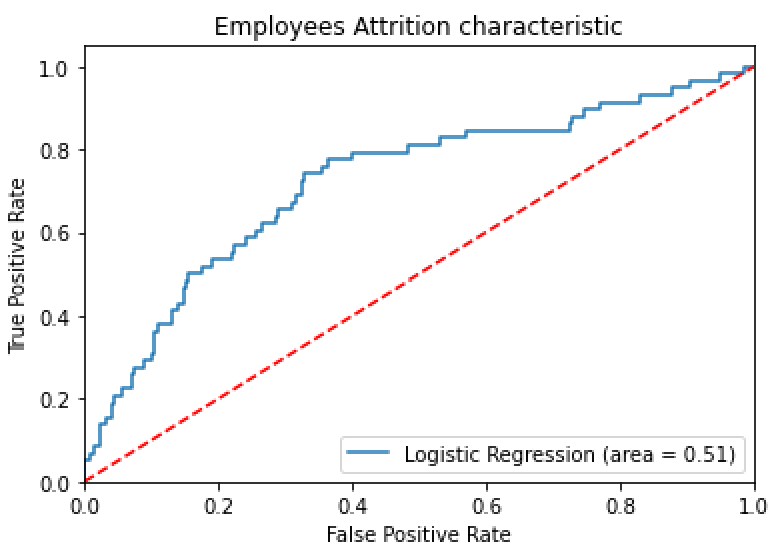
In [25]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, predicted_y)
```

Out[25]:
```
array([[310,   0],
       [ 57,   1]], dtype=int64)
```

In [26]:
```
#mean absolute error
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, predicted_y)
```

Out[26]: 39.49728260869565

In [27]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Employees Attrition characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



3. 2. Decision Tree Algorithm

In [28]:
```python
# trying DecisionTree Classifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.tree import plot_tree  # Tree plotting

from sklearn.metrics import accuracy_score


DTC = DecisionTreeClassifier(random_state = 100,max_depth=3)
DTC.fit(X_train,y_train)
```

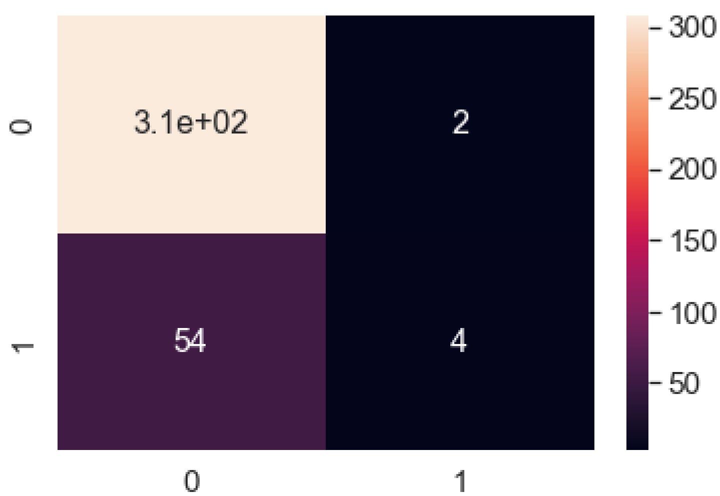
Out[28]:
```
DecisionTreeClassifier(max_depth=3, random_state=100)
```

In [29]:
```python
predicted_DTC = DTC.predict(X_test)

conf_mat = confusion_matrix(y_test, predicted_DTC)
conf_mat
```

Out[29]:
```
array([[308,   2],
       [ 54,   4]], dtype=int64)
```

In [30]:
```python
model = DecisionTreeClassifier(random_state = 100,max_depth=3)
```

In [31]:
```python
df_cm = pd.DataFrame(conf_mat)
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size

plt.show()
```

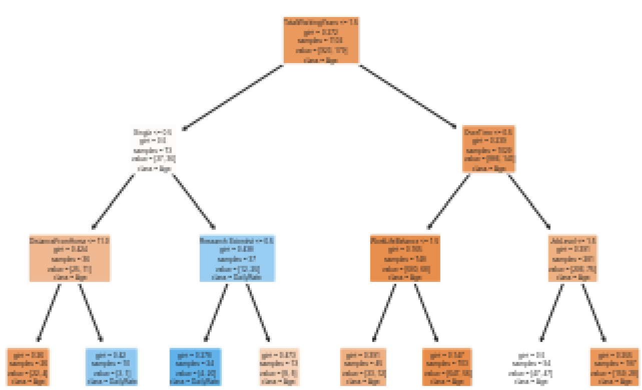


In [32]:
```python
ac2 = accuracy_score(y_test,predicted_DTC)
ac2
```

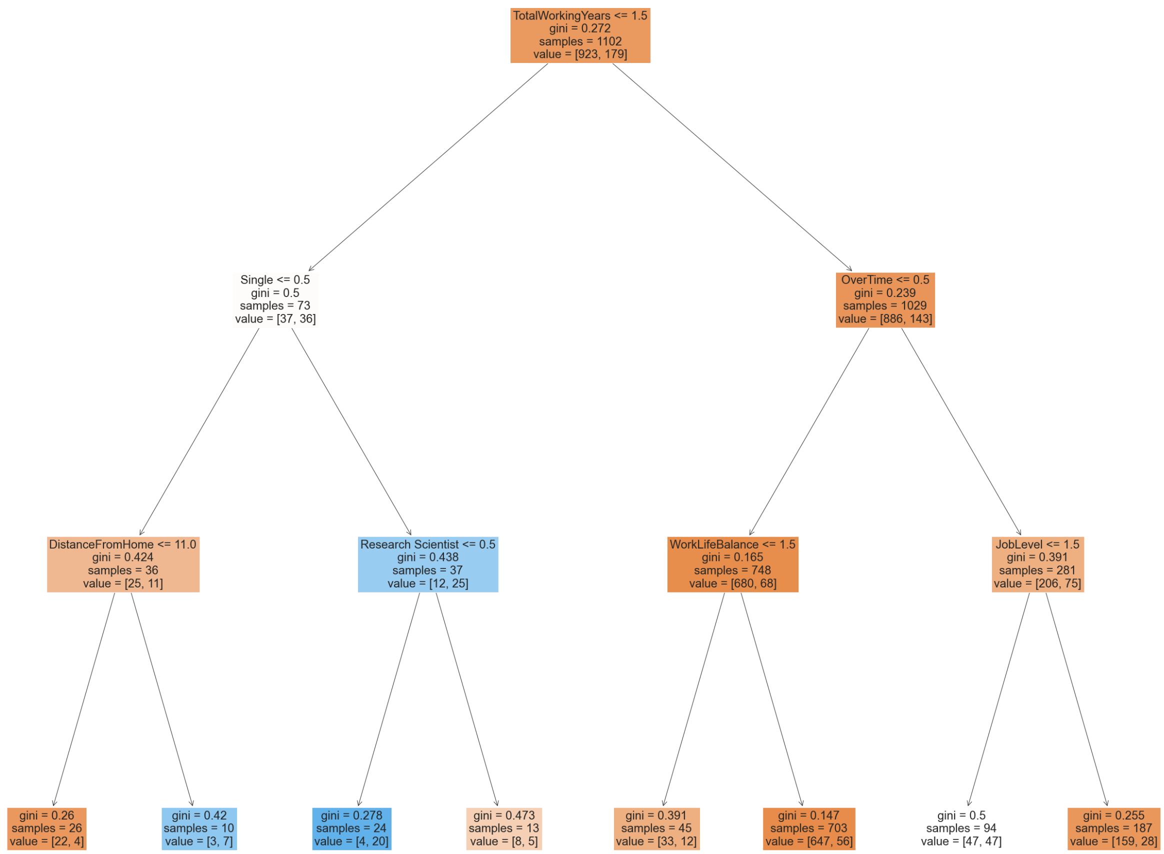
Out[32]:
```
0.8478260869565217
```

In [33]:
```python
cols = list(X_train.columns)
plot_tree(DTC,
          feature_names = cols,
          class_names = cols,
          filled = True,
          rounded = True)
```

Out[33]:
```
[Text(167.4, 190.26, 'TotalWorkingYears <= 1.5\ngini = 0.272\nsamples = 1102\nv
alue = [923, 179]\nclass = Age'),
 Text(83.7, 135.9, 'Single <= 0.5\ngini = 0.5\nsamples = 73\nvalue = [37, 36]\n
class = Age'),
 Text(41.85, 81.53999999999999, 'DistanceFromHome <= 11.0\ngini = 0.424\nsample
s = 36\nvalue = [25, 11]\nclass = Age'),
 Text(20.925, 27.180000000000007, 'gini = 0.26\nsamples = 26\nvalue = [22, 4]\n
class = Age'),
 Text(62.775000000000006, 27.180000000000007, 'gini = 0.42\nsamples = 10\nvalue
= [3, 7]\nclass = DailyRate'),
 Text(125.55000000000001, 81.53999999999999, 'Research Scientist <= 0.5\ngini =
0.438\nsamples = 37\nvalue = [12, 25]\nclass = DailyRate'),
 Text(104.625, 27.180000000000007, 'gini = 0.278\nsamples = 24\nvalue = [4, 20]
\nclass = DailyRate'),
 Text(146.475, 27.180000000000007, 'gini = 0.473\nsamples = 13\nvalue = [8, 5]
\nclass = Age'),
 Text(251.10000000000002, 135.9, 'OverTime <= 0.5\ngini = 0.239\nsamples = 1029
\nvalue = [886, 143]\nclass = Age'),
 Text(209.25, 81.53999999999999, 'WorkLifeBalance <= 1.5\ngini = 0.165\nsamples
= 748\nvalue = [680, 68]\nclass = Age'),
 Text(188.32500000000002, 27.180000000000007, 'gini = 0.391\nsamples = 45\nvalu
e = [33, 12]\nclass = Age'),
 Text(230.175, 27.180000000000007, 'gini = 0.147\nsamples = 703\nvalue = [647,
56]\nclass = Age'),
 Text(292.95, 81.53999999999999, 'JobLevel <= 1.5\ngini = 0.391\nsamples = 281
\nvalue = [206, 75]\nclass = Age'),
 Text(272.02500000000003, 27.180000000000007, 'gini = 0.5\nsamples = 94\nvalue
= [47, 47]\nclass = Age'),
 Text(313.875, 27.180000000000007, 'gini = 0.255\nsamples = 187\nvalue = [159,
28]\nclass = Age')]
```



In [34]:
```python
fig = plt.figure(figsize=(45,40))
_ = tree.plot_tree(DTC,
                   feature_names=cols,
                   filled=True)
fig.savefig("decistion_tree_RatingsPred.png")
```



3. 3. Random Forest Classification

In [35]:
```python
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier(n_estimators = 100, criterion = 'entropy', random_st
```

In [36]:
```python
RFC.fit(X_train, y_train)
```

Out[36]:
```
RandomForestClassifier(criterion='entropy', random_state=0)
```

In [37]:
```
#Accuracy on training data
RFC.score(X_test, y_test)
```

Out[37]: 0.8614130434782609

In [38]:
```
predicted_RFC = RFC.predict(X_test)
```

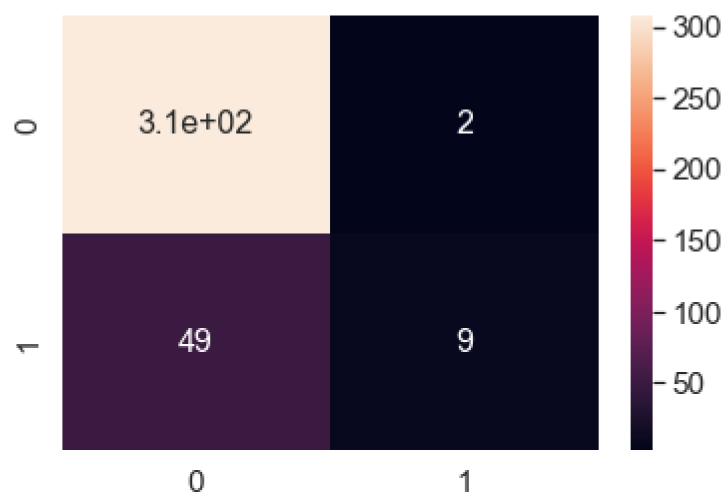
In [39]:
```
#mean absolute error
mean_absolute_error(y_test, predicted_RFC)
```

Out[39]: 33.95923913043478

In [40]:
```
#Confusion Matrix
cm = confusion_matrix(y_test, predicted_RFC)
cm
```

Out[40]:
```
array([[308,   2],
       [ 49,   9]], dtype=int64)
```

In [41]:
```
df_cm = pd.DataFrame(cm)
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size

plt.show()
```



In [42]:
```
from sklearn.metrics import accuracy_score
ac1 = accuracy_score(y_test,predicted_RFC)
ac1
```

Out[42]: 0.8614130434782609

3. 4. Support Vector Machines

In [43]:
```
from sklearn import svm
SVM = svm.SVC(kernel='linear')
```

In [44]:
```
SVM.fit(X_train,y_train)
```

Out[44]:
```
SVC(kernel='linear')
```

In [45]:
```
predicted_SVM = SVM.predict(X_test)
```

In [46]:
```
from sklearn.metrics import accuracy_score
ac5 = accuracy_score(y_test,predicted_SVM)
ac5
```

Out[46]:
```
0.8505434782608695
```

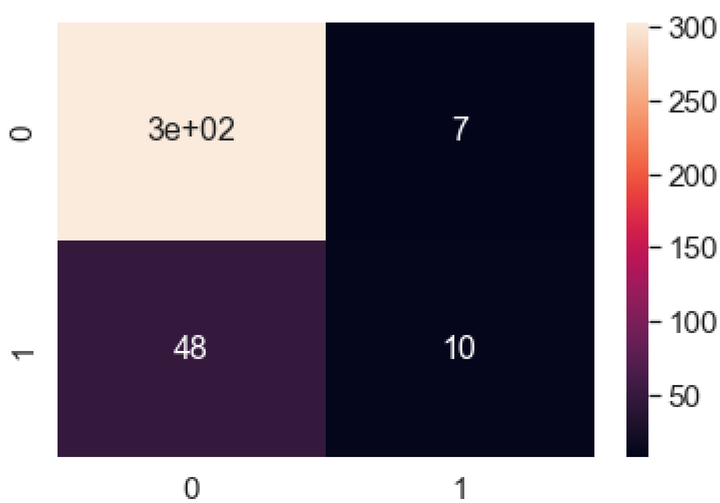
In [47]:
```
#mean absolute error
mean_absolute_error(y_test, predicted_SVM)
```

Out[47]:
```
33.27989130434783
```

In [48]:
```
cm_ = confusion_matrix(y_test, predicted_SVM)
cm_
```

Out[48]:
```
array([[303,   7],
       [ 48,  10]], dtype=int64)
```

In [49]:
```
df_cm = pd.DataFrame(cm_)
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size

plt.show()
```



3. 5. Neural Networks

In [51]:
```python
# X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.33)
X_train.shape, X_test.shape
```

Out[51]:
```
((984, 44), (486, 44))
```

In [52]:
```python
#Dependencies
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score

# Neural network
model = Sequential([
    Dense(64, activation='relu', input_dim=X_train.shape[1]),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')])

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=100, batch_size=64)
```

```
Epoch 1/100
16/16 [==============================] - 0s 1ms/step - loss: 421.4622 - accu
racy: 0.6839
Epoch 2/100
16/16 [==============================] - 0s 1ms/step - loss: 286.0669 - accu
racy: 0.7307
Epoch 3/100
16/16 [==============================] - 0s 1ms/step - loss: 232.2785 - accu
racy: 0.7470
Epoch 4/100
16/16 [==============================] - 0s 1ms/step - loss: 203.4974 - accu
racy: 0.7205
Epoch 5/100
16/16 [==============================] - 0s 1ms/step - loss: 193.9079 - accu
racy: 0.7185
Epoch 6/100
16/16 [==============================] - 0s 2ms/step - loss: 142.4707 - accu
racy: 0.7449
Epoch 7/100
```

In [53]:
```python
y_pred = model.predict(X_test)
#Converting predictions to label
pred = list()
for i in range(len(y_pred)):
    pred.append(np.round(y_pred[i]))
```

In [54]:
```python
ac6 = accuracy_score(pred,y_test)
print('Accuracy is:', ac6 * 100)
```

```
Accuracy is: 84.36213991769547
```

In [55]:
```python
def adj_r2_score(r2, n, k):
    return 1-((1-r2)*((n-1)/(n-k-1)))

r2_test = r2_score(y_test, y_pred)
print("R-squared is: %f"%r2_test)
```

```
R-squared is: -0.075872
```

In [56]:
```python
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred)
```

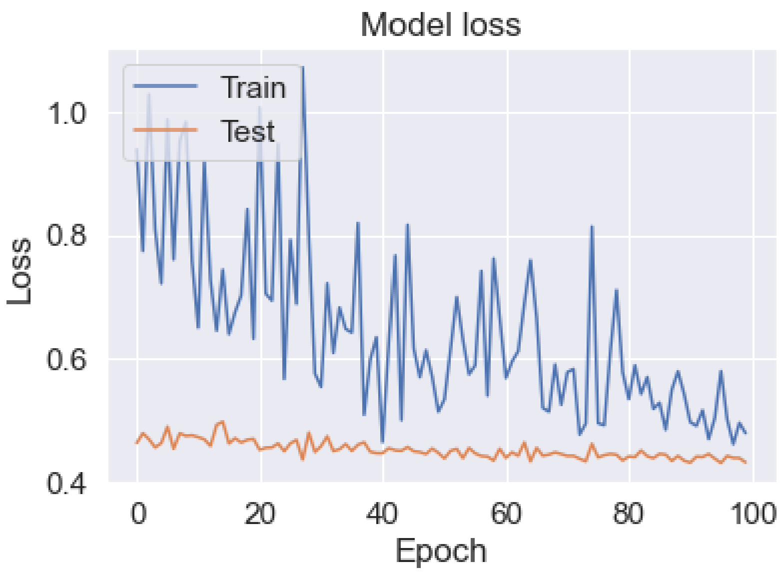
Out[56]:
```
0.3361135
```

In [57]:
```python
history = model.fit(X_train, y_train,validation_data = (X_test,y_test), epochs=10
```

```
Epoch 1/100
16/16 [==============================] - 0s 13ms/step - loss: 0.9393 - accur
acy: 0.8252 - val_loss: 0.4632 - val_accuracy: 0.8436
Epoch 2/100
16/16 [==============================] - 0s 3ms/step - loss: 0.7740 - accura
cy: 0.8283 - val_loss: 0.4792 - val_accuracy: 0.8436
Epoch 3/100
16/16 [==============================] - 0s 3ms/step - loss: 1.0285 - accura
cy: 0.8283 - val_loss: 0.4692 - val_accuracy: 0.8436
Epoch 4/100
16/16 [==============================] - 0s 3ms/step - loss: 0.8093 - accura
cy: 0.8201 - val_loss: 0.4560 - val_accuracy: 0.8436
Epoch 5/100
16/16 [==============================] - 0s 3ms/step - loss: 0.7214 - accura
cy: 0.8262 - val_loss: 0.4631 - val_accuracy: 0.8436
Epoch 6/100
16/16 [==============================] - 0s 3ms/step - loss: 0.9875 - accura
cy: 0.8222 - val_loss: 0.4896 - val_accuracy: 0.8436
Epoch 7/100
```

In [58]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

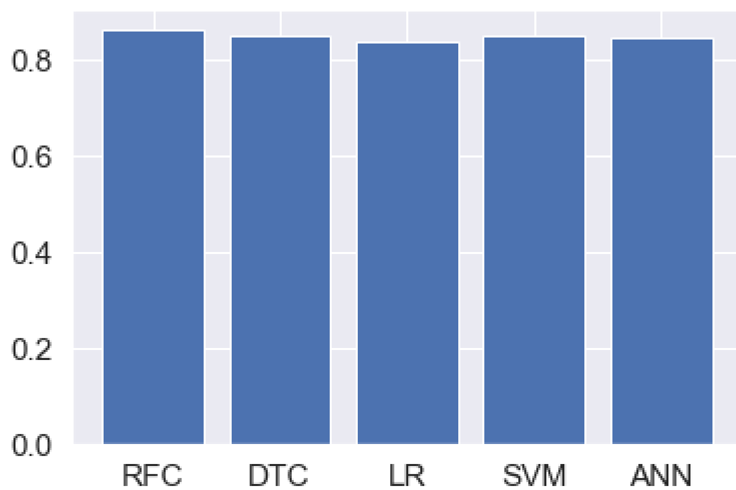


4. Compare various models and their accuracy

In [59]:
```
r= {}
r['RFC']=ac1
r['DTC']=ac2
r['LR']= ac3
r['SVM']=ac5
r['ANN']=ac6
print(r)
plt.bar(range(len(r)), list(r.values()), align='center')
plt.xticks(range(len(r)), list(r.keys()))
```

```
{'RFC': 0.8614130434782609, 'DTC': 0.8478260869565217, 'LR': 0.838475499092559,
'SVM': 0.8505434782608695, 'ANN': 0.8436213991769548}
```

Out[59]:
```
([<matplotlib.axis.XTick at 0x1cad1850be0>,
  <matplotlib.axis.XTick at 0x1cad1850bb0>,
  <matplotlib.axis.XTick at 0x1cad18501c0>,
  <matplotlib.axis.XTick at 0x1cad1884cd0>,
  <matplotlib.axis.XTick at 0x1cad1890460>],
 [Text(0, 0, 'RFC'),
  Text(1, 0, 'DTC'),
  Text(2, 0, 'LR'),
  Text(3, 0, 'SVM'),
  Text(4, 0, 'ANN')])
```



The accuracies of various models are as follows:

- **Logistic Regression - 84.51%**
- **Random Forests - 86.14%**
- **Decision Tree - 84.78%**
- **Support Vector Machines - 85.05%**
- **Aritifical Neural Network - 83.53%**

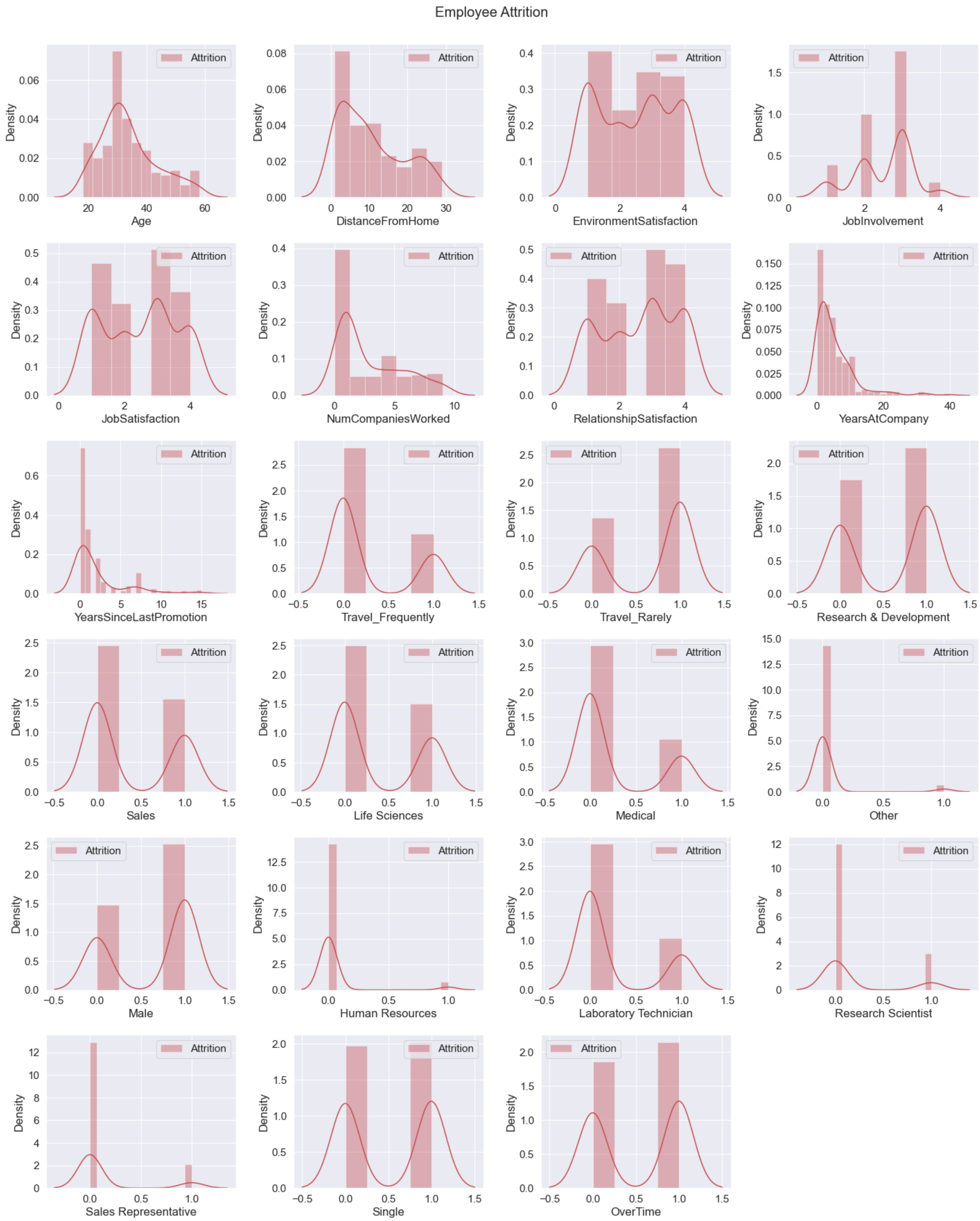
5. Feature Selection & P-value

In [60]:
```python
from sklearn.preprocessing import LabelEncoder
import statsmodels.api as sm
import warnings
warnings.filterwarnings("ignore")
```

In [61]:
```python
def backwardElimination(x, Y, sl, columns):
    numVars = len(x[0])
    for i in range(0, numVars):
        regressor_OLS = sm.OLS(Y, x).fit()

        maxVar = max(regressor_OLS.pvalues).astype(float)
        if maxVar > sl:
            for j in range(0, numVars - i):
                if (regressor_OLS.pvalues[j].astype(float) == maxVar):
                    x = np.delete(x, j, 1)
                    columns = np.delete(columns, j)

    regressor_OLS.summary()
    return x, columns
```

In [62]:
```
def featureSelection(enc_df):

    enc_df = enc_df[ ['Attrition'] + [ col for col in enc_df.columns if col != 'A

    label_encoder = LabelEncoder()
    enc_df.iloc[:,0] = label_encoder.fit_transform(enc_df.iloc[:,0]).astype('floa

    corr = enc_df.corr()

    columns = np.full((corr.shape[0],), True, dtype=bool)

    for i in range(corr.shape[0]):
        for j in range(i+1, corr.shape[0]):
            if corr.iloc[i,j] >= 0.7:
                if columns[j]:
                    columns[j] = False

    # print(columns)
    selected_columns = enc_df.columns[columns]

    enc_df = enc_df[selected_columns]

    selected_columns = selected_columns[1:]

    SL = 0.05
    data_modeled, selected_columns = backwardElimination(enc_df.iloc[:,1:].values
    result = pd.DataFrame()
    result['Attrition'] = enc_df.iloc[:,0]

    data = pd.DataFrame(data = data_modeled, columns = selected_columns)

    fig = plt.figure(figsize = (20, 25))
    j = 0
    for i in data.columns:
        plt.subplot(6, 4, j+1)
        j += 1
        sns.distplot(data[i][result['Attrition']==1], color='r', label = 'Attriti
        plt.legend(loc='best')
    fig.suptitle('Employee Attrition')
    fig.tight_layout()
    fig.subplots_adjust(top=0.95)
    plt.show()
```

In [63]:
```
featureSelection(oh_en_df)
```



6. Hypothesis Testing

Let's go back to our hypothesis.

Null Hypothesis People who are satisfied with their job are less likely to leave the company.

Alternate Hypothesis People who are satisfied with their job are more likely to leave the company.

Chi-Square Test

In [64]:
```
from scipy.stats import chi2_contingency

def chiSquareTest(expected_dependent_column: str):
    data = [oh_en_df[expected_dependent_column], oh_en_df['Attrition']]
    stat, p, dof, expected = chi2_contingency(data)

    alpha = 0.05
    print("p value is " + str(p))
    if p <= alpha:
        print('Attrition is dependent on', expected_dependent_column, '(Hypothesi
    else:
        print('Attrition is Independent of', expected_dependent_column , '(Reject
```

In [65]:
```
chiSquareTest('MonthlyIncome')

chiSquareTest('JobSatisfaction')
```

```
p value is 1.85346467060735e-80
Attrition is dependent on MonthlyIncome (Hypothesis holds true)

p value is 0.999999802750287
Attrition is Independent of JobSatisfaction (Reject Hypothesis)
```

7. Conclusion

Based on the Chi-square test and the p-values, we can say that **our hypothesis doesn't hold true**.

On the contrary to popular notions that you will continue to work at a job that satisfies you, we have proved otherwise for this data.

Obviously, since this is secondary data, we have no way of identifying whether the employees are indeed satisfied.

Some factors that are significant in to the attrition of an employee is how long they have worked at this company `YearsAtCompany` , their monthly pay `MonthlyIncome` ,environment satisfacation `EnvironmentSatisfaction` , whether or not they work overtime `Overtime` and number of other companies they worked at `NumCompaniesWorked` .

Variables that are independent determining whether an employee stays or not are `JobSatisfaction` , `Education` , `Gender` , `PerformanceRating` .

We hope that we receive the following "Dua Lipa" reaction for this project!!

Thank You


YOU GET A THANK YOU