assignment-4 September 27, 2023

0.1 Assignment - 4

0.2 RAVADAGUNDI SANATH KUMAR

sanathkumar.21bce9610@vitapstudent.ac.in (mailto:sanathkumar.21bce9610@vitapstudent.ac.in)

0.3 • Data Preprocessing.

```
o Import the Libraries.
o Importing the dataset.
o Checking for Null Values.
o Data Visualization.
o Outlier Detection
o Splitting Dependent and Independent variables
o- Encoding
o Feature Scaling.
o Splitting Data into Train and Test.
```

0.3.1 Import the Libraries.

In [72]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

0.3.2 Importing the dataset.

In [73]:
```
df=pd.read_csv("C:/Users/rsana/Downloads/archive/WA_Fn-UseC_-HR-Employee-Attr
```

In [74]:
```
df.head()
```

Out[74]:

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

5 rows × 35 columns

In [75]:
```
df.shape
```

Out[75]:
```
(1470, 35)
```

In [76]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

In [77]:

```
df.describe()
```

Out[77]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeN
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.(
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.8
std	9.135373	403.509100	8.106864	1.024165	0.0	602.(
min	18.000000	102.000000	1.000000	1.000000	1.0	1.(
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.2
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.5
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.7
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.(

8 rows × 26 columns

0.3.3 Checking for null values

In [78]:
```
df.isnull().any()
```

Out[78]:
```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              False
EmployeeCount               False
EmployeeNumber              False
EnvironmentSatisfaction     False
Gender                      False
HourlyRate                  False
JobInvolvement              False
JobLevel                    False
JobRole                     False
JobSatisfaction             False
MaritalStatus               False
MonthlyIncome               False
MonthlyRate                 False
NumCompaniesWorked          False
Over18                      False
OverTime                    False
PercentSalaryHike           False
PerformanceRating           False
RelationshipSatisfaction    False
StandardHours               False
StockOptionLevel            False
TotalWorkingYears           False
TrainingTimesLastYear       False
WorkLifeBalance             False
YearsAtCompany              False
YearsInCurrentRole          False
YearsSinceLastPromotion     False
YearsWithCurrManager        False
dtype: bool
```

In [79]:
```
df.isnull().sum()
```

Out[79]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [80]:
```
import warnings
# Ignore the specified warnings globally
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
```

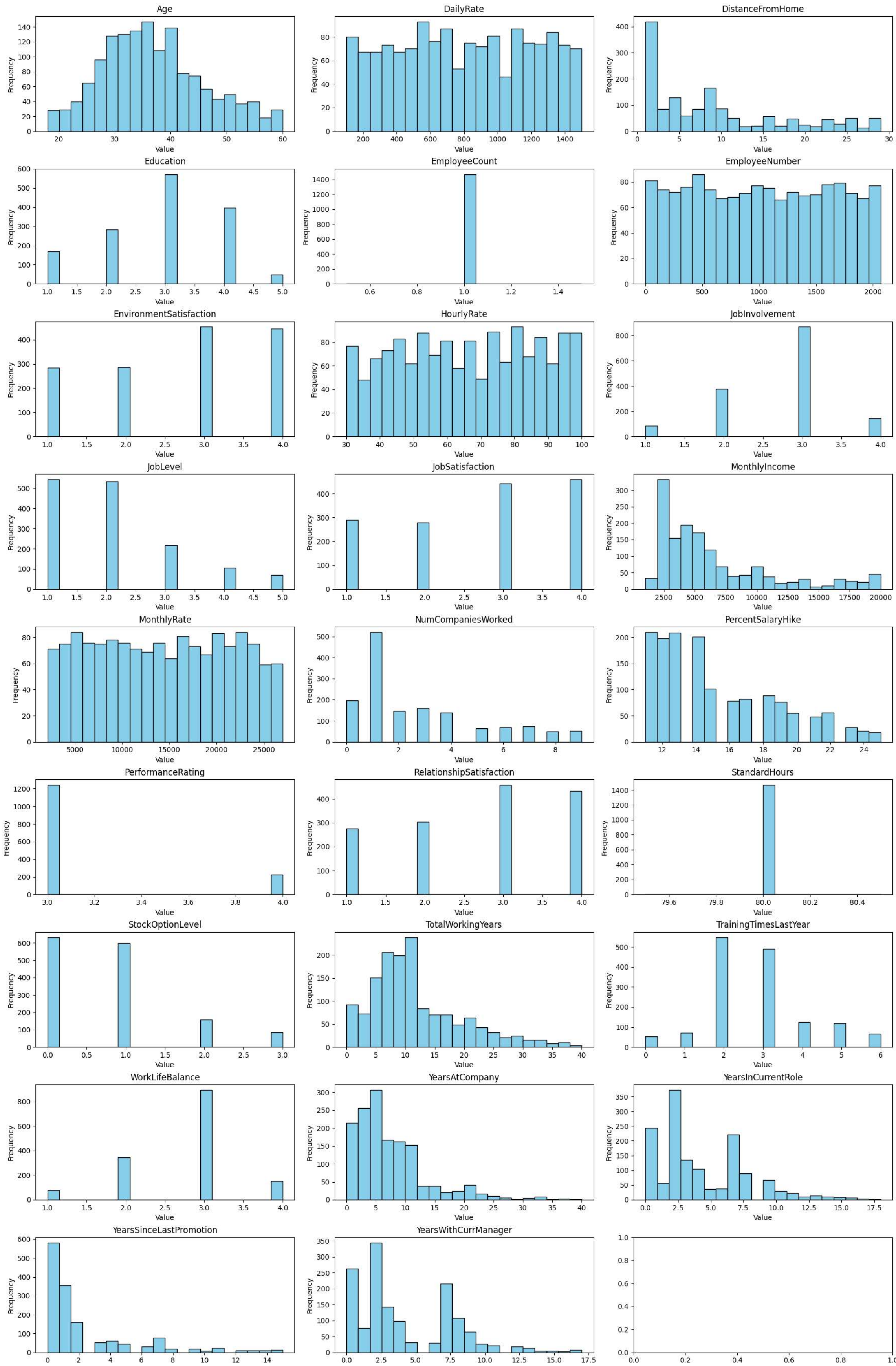
0.3.4 Data Visualization

In [81]:

```
numerical_vars = df.select_dtypes(include=['int64', 'float64']).columns.tolis
# Create histograms for the selected numerical variables
fig, axes = plt.subplots(nrows=9, ncols=3, figsize=(18, 30))
fig.suptitle('Distribution of Numerical Variables', fontsize=16)
for ax, var in zip(axes.flatten(), numerical_vars):
  ax.hist(df[var], bins=20, color='skyblue', edgecolor='black')
  ax.set_title(var)
  ax.set_xlabel('Value')
  ax.set_ylabel('Frequency')

plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()
```

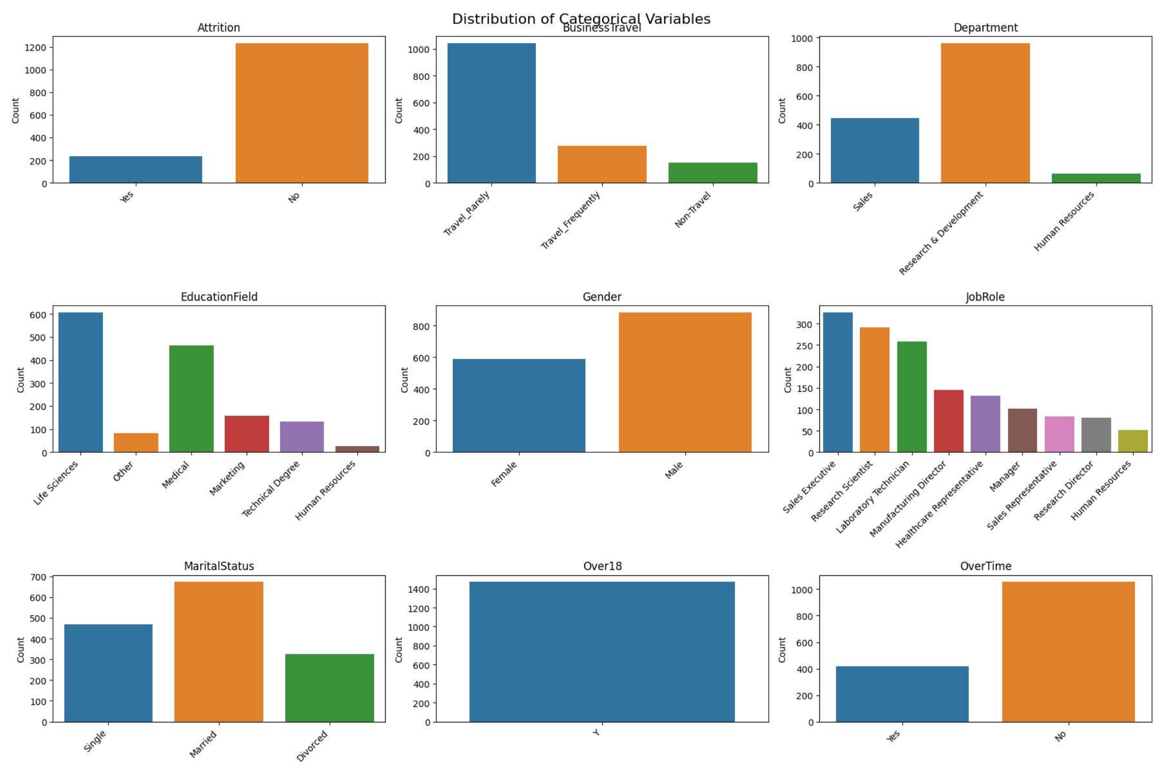
Distribution of Numerical Variables

Distribution of Numerical Variables

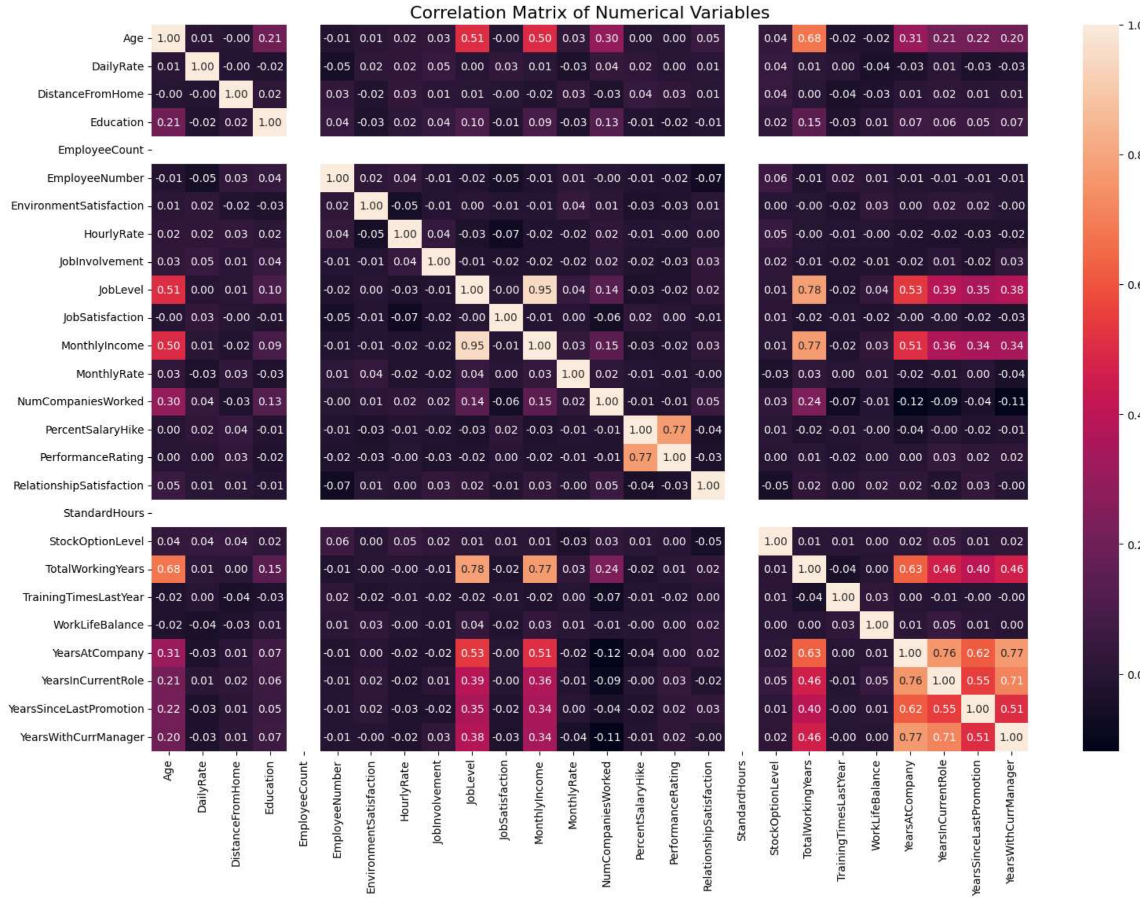


In [82]:

```
# Select categorical variables for visualization
categorical_vars = df.select_dtypes(include=['object']).columns.tolist()
# Create bar plots for the selected categorical variables
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(18, 12))
fig.suptitle('Distribution of Categorical Variables', fontsize=16)
for ax, var in zip(axes.flatten(), categorical_vars):
 sns.countplot(data=df, x=var, ax=ax)
 ax.set_title(var)
 ax.set_xlabel('')
 ax.set_ylabel('Count')
 ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
# Remove the empty subplots
for ax in axes.flatten()[len(categorical_vars):]:
 fig.delaxes(ax)
plt.tight_layout()
plt.subplots_adjust(top=0.95)
plt.show()
```



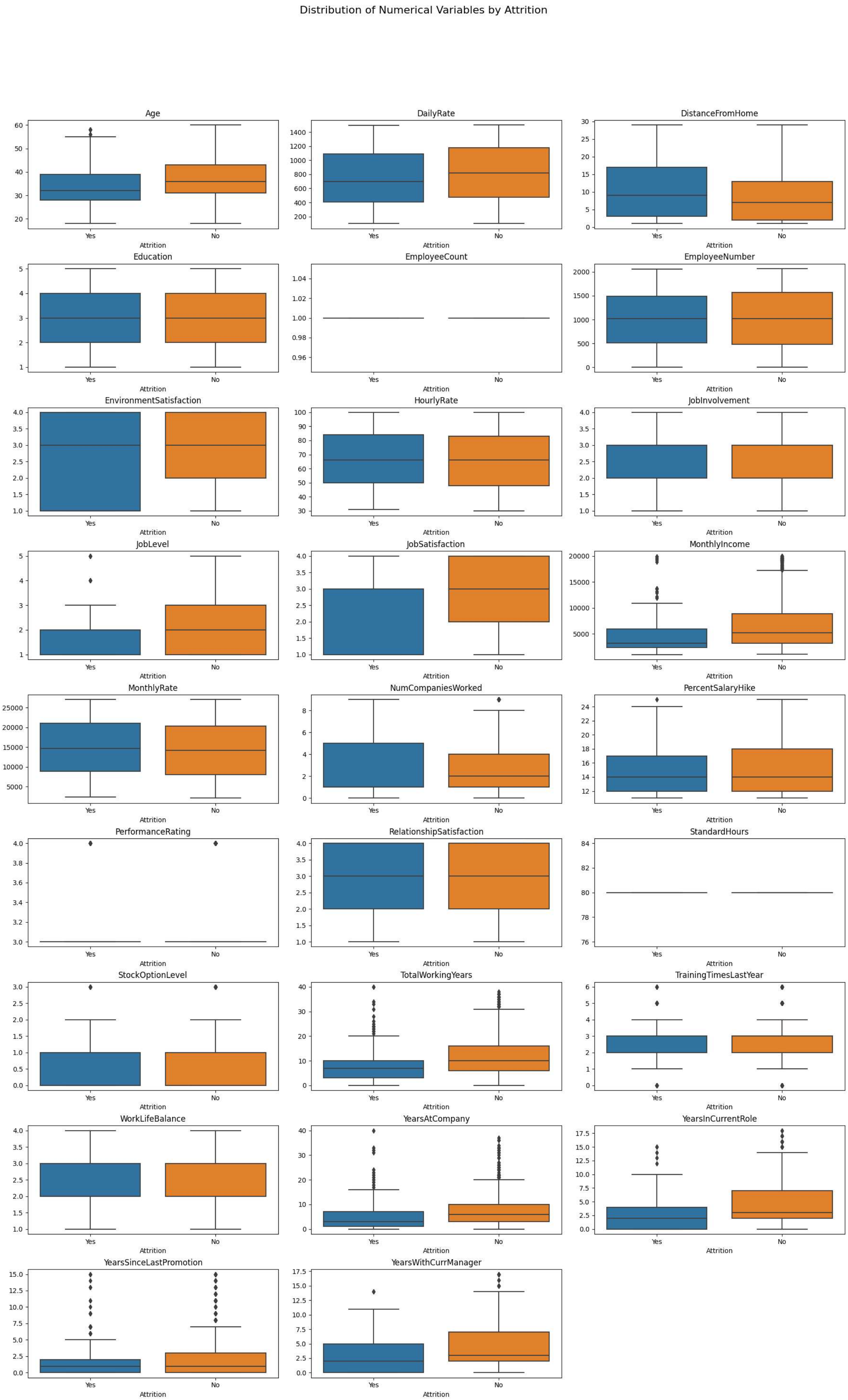
In [83]:
```
# Compute the correlation matrix
correlation_matrix = df.corr(numeric_only=True)
# Plot the heatmap of the correlation matrix
plt.figure(figsize=(18, 12))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f')
plt.title('Correlation Matrix of Numerical Variables', fontsize=16)
plt.show()
```



In [84]:

```
# Select numerical variables for visualization
numerical_vars_to_compare = df.select_dtypes(include=['int64', 'float64']).co

# Create box plots for the selected numerical variables grouped by Attrition
fig, axes = plt.subplots(nrows=9, ncols=3, figsize=(18, 30))
fig.suptitle('Distribution of Numerical Variables by Attrition', fontsize=16)

for ax, var in zip(axes.flatten(), numerical_vars_to_compare):
 sns.boxplot(data=df, x='Attrition', y=var, ax=ax)
 ax.set_title(var)
 ax.set_xlabel('Attrition')
 ax.set_ylabel('')

# Remove the empty subplot
fig.delaxes(axes.flatten()[-1])
plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()
```

Distribution of Numerical Variables by Attrition


Distribution of Numerical Variables by Attrition
Age
DailyRate
DistanceFromHome
Education
EmployeeCount
EmployeeNumber
EnvironmentSatisfaction
HourlyRate
JobInvolvement
JobLevel
JobSatisfaction
MonthlyIncome
MonthlyRate
NumCompaniesWorked
PercentSalaryHike
PerformanceRating
RelationshipSatisfaction
StandardHours
StockOptionLevel
TotalWorkingYears
TrainingTimesLastYear
WorkLifeBalance
YearsAtCompany
YearsInCurrentRole
YearsSinceLastPromotion
YearsWithCurrManager
Attrition
Yes
No

0.3.5 Splitting Dependent and Independent variables

In [85]:
```
x = df.drop(columns=['Attrition'],axis=1)
y = df['Attrition']
```

In [86]:
```
x.shape,y.shape
```

Out[86]:
```
((1470, 34), (1470,))
```

In [87]:
```
y.head()
```

Out[87]:
```
0    Yes
1     No
2    Yes
3     No
4     No
Name: Attrition, dtype: object
```

0.3.6 Encoding

In [88]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [89]:
```
y=le.fit_transform(y)
```

In [90]:
```
x["BusinessTravel"]=le.fit_transform(x["BusinessTravel"])
x["Department"]=le.fit_transform(x["Department"])
x["EducationField"]=le.fit_transform(x["EducationField"])
x["Gender"]=le.fit_transform(x["Gender"])
x["JobRole"]=le.fit_transform(x["JobRole"])
x["MaritalStatus"]=le.fit_transform(x["MaritalStatus"])
x["Over18"]=le.fit_transform(x["Over18"])
x["OverTime"]=le.fit_transform(x["OverTime"])
```

In [91]:
```
x.head()
```

Out[91]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField
0	41	2	1102	2	1	2	1
1	49	1	279	1	8	1	1
2	37	2	1373	1	2	2	4
3	33	1	1392	1	3	4	1
4	27	2	591	1	2	1	3

5 rows × 34 columns

0.3.7 Feature Scaling

In [92]:
```
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
```

In [93]:
```
x_scaled = pd.DataFrame(ms.fit_transform(x), columns = x.columns)
x_scaled.head()
```

Out[93]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationF
0	0.547619	1.0	0.715820	1.0	0.000000	0.25	
1	0.738095	0.5	0.126700	0.5	0.250000	0.00	
2	0.452381	1.0	0.909807	0.5	0.035714	0.25	
3	0.357143	0.5	0.923407	0.5	0.071429	0.75	
4	0.214286	1.0	0.350036	0.5	0.035714	0.00	

5 rows × 34 columns

0.3.8 Spliting into Training and Testing dataset

In [94]:
```
from sklearn.model_selection import train_test_split
```

In [95]:
```
x_train1, x_test1, y_train1, y_test1 = train_test_split(x, y,
test_size = 0.3, random_state = 0)
```

In [96]:
```
print(x_train1.shape)
print(x_test1.shape)
print(y_train1.shape)
print(y_test1.shape)
```

```
(1029, 34)
(441, 34)
(1029,)
(441,)
```

In [97]:
```
x_train2, x_test2, y_train2, y_test2 = train_test_split(x_scaled, y, test_size = 0.3, random_state = 0)
```

In [98]:
```
print(x_train2.shape)
print(x_test2.shape)
print(y_train2.shape)
print(y_test2.shape)
```

```
(1029, 34)
(441, 34)
(1029,)
(441,)
```

0.4 Model Building

```
1.Import the model building Libraries
2.Initializing the model
3.Training and testing the model
4.Evaluation of Model
5.Save the Model
```

0.4.1 Logistic Regression

In [99]:
```
from sklearn.linear_model import LogisticRegression
lrmodel = LogisticRegression()
```

In [100]:
```
lrmodel.fit(x_train2,y_train2)
```

Out[100]:
```
▾ LogisticRegression
LogisticRegression()
```

In [101]:
```
predlr=lrmodel.predict(x_test2)
```

In [102]:
```
from sklearn.metrics import accuracy_score, confusion_matrix,classification_r
```

In [103]:
```
 accuracy_score(y_test2,predlr)
```

Out[103]: 0.8866213151927438

In [104]:
```
confusion_matrix(y_test2,predlr)
```

Out[104]:
```
array([[368,   3],
       [ 47,  23]], dtype=int64)
```

In [105]:
```
pd.crosstab(y_test2,predlr)
```

Out[105]:

col_0	0	1
row_0		
0	368	3
1	47	23

In [106]:
```
print(classification_report(y_test2,predlr))
```

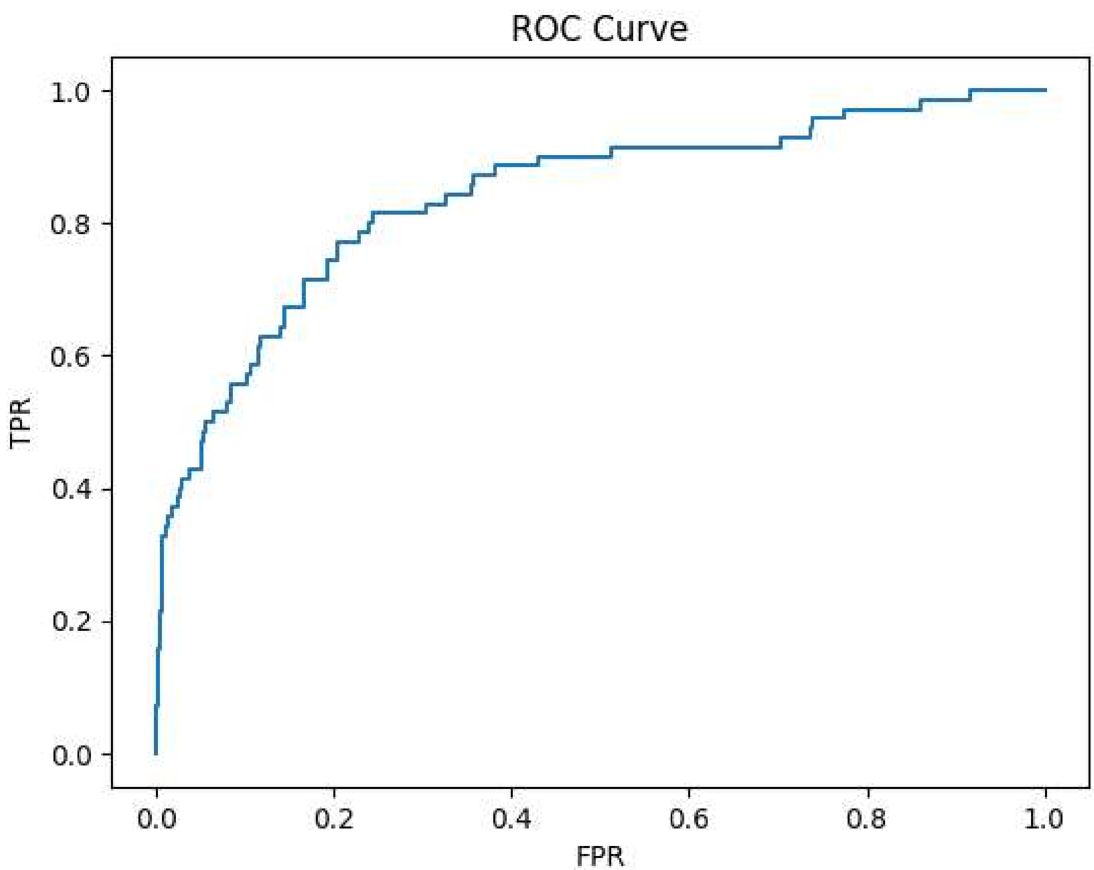
```
              precision    recall  f1-score   support

           0       0.89      0.99      0.94       371
           1       0.88      0.33      0.48        70

    accuracy                           0.89       441
   macro avg       0.89      0.66      0.71       441
weighted avg       0.89      0.89      0.86       441
```

In [107]:
```
roc_auc_score(y_test2,predlr)
```

Out[107]: 0.66024258760107 81

In [108]:
```
prob1 = lrmodel.predict_proba(x_test2)[:,1]
fpr1, tpr1, thres1 = roc_curve( y_test2, prob1)
```

In [109]:
```
plt.plot(fpr1, tpr1)
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.show()
```



0.4.2 Decision Tree

In [110]:
```
from sklearn.tree import DecisionTreeClassifier
dtmodel=DecisionTreeClassifier()
```

In [111]:
```
dtmodel.fit(x_train1, y_train1)
```

Out[111]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [112]:
```
preddt = dtmodel.predict(x_test1)
```

In [113]:
```
accuracy_score(y_test1,preddt)
```

Out[113]:
```
0.764172335600907
```

In [114]:
```
confusion_matrix(y_test1,preddt)
```

Out[114]:
```
array([[315,  56],
       [ 48,  22]], dtype=int64)
```

In [115]:
```
pd.crosstab(y_test1,preddt)
```

Out[115]:

col_0	0	1
row_0		
0	315	56
1	48	22

In [116]:
```
print(classification_report(y_test1,preddt))
```

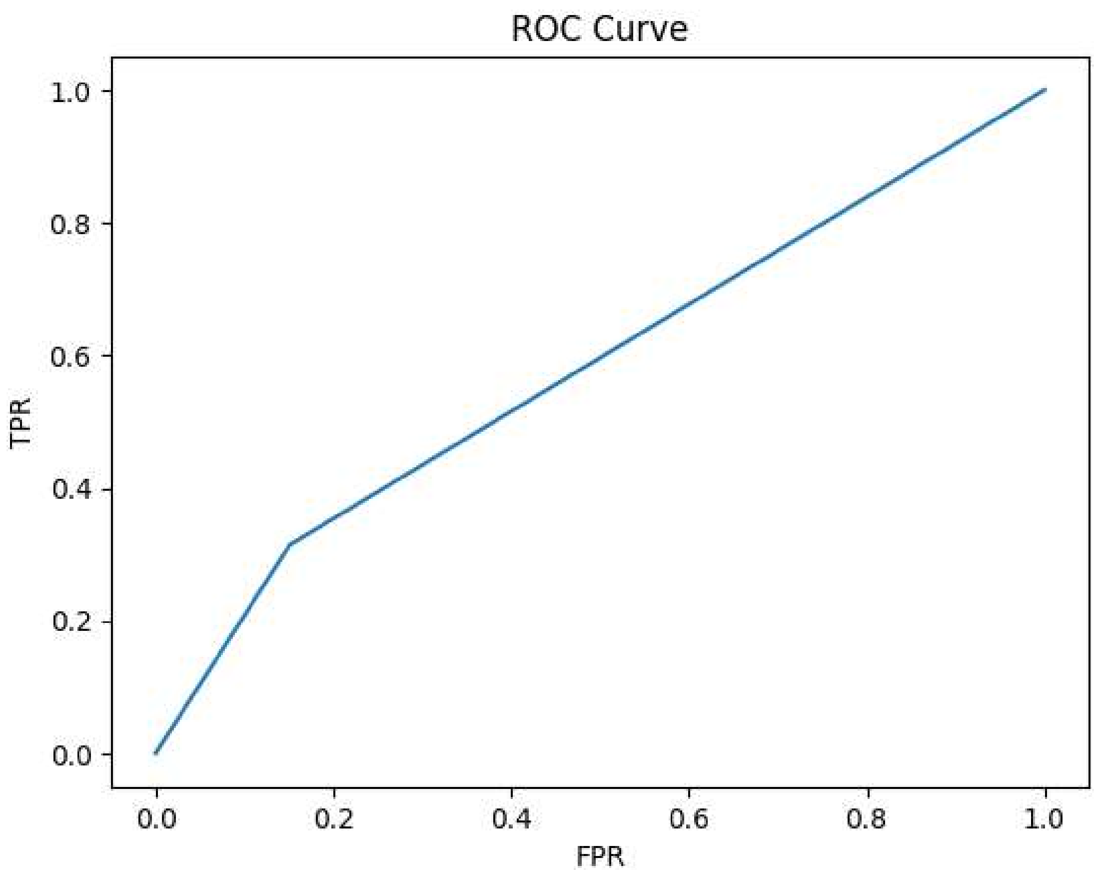
```
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       371
           1       0.28      0.31      0.30        70

    accuracy                           0.76       441
   macro avg       0.57      0.58      0.58       441
weighted avg       0.77      0.76      0.77       441
```

In [117]:
```
roc_auc_score(y_test1, preddt)
```

Out[117]:
```
0.5816711590296496
```

In [118]:
```
prob2 = dtmodel.predict_proba(x_test1)[:,1]
fpr2, tpr2, thres2 = roc_curve( y_test1, prob2)
```

In [119]:
```
plt.plot(fpr2, tpr2)
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.show()
```



0.4.3 Random Forest

In [120]:
```
from sklearn.ensemble import RandomForestClassifier
rfmodel = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [121]:
```
rfmodel.fit(x_train1, y_train1)
```

Out[121]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [122]:
```
predrf=rfmodel.predict(x_test1)
accuracy_score(y_test1,predrf)
```

Out[122]: 0.8616780045351474

In [123]:
```
confusion_matrix(y_test1,predrf)
```

Out[123]:
```
array([[367,   4],
       [ 57,  13]], dtype=int64)
```

In [124]:
```
pd.crosstab(y_test1,predrf)
```

Out[124]:

col_0	0	1
row_0		
0	367	4
1	57	13

In [125]:
```
print(classification_report(y_test1,predrf))
```

```
              precision    recall  f1-score   support

           0       0.87      0.99      0.92       371
           1       0.76      0.19      0.30        70

    accuracy                           0.86       441
   macro avg       0.82      0.59      0.61       441
weighted avg       0.85      0.86      0.82       441
```

0.4.4 Accuracy of All Three Models

In [126]:
```
print("Accuracy of Logistic Regression :", accuracy_score(y_test2,predlr))
print("Accuracy of Decision Tree Classifier :", accuracy_score(y_test1,preddt
print("Accuracy of Random Forest Classifier :", accuracy_score(y_test1,predrf
```

```
Accuracy of Logistic Regression : 0.8866213151927438
Accuracy of Decision Tree Classifier : 0.764172335600907
Accuracy of Random Forest Classifier : 0.8616780045351474
```