In [54]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [55]:
```
df = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

In [63]:
```
df.head()
```

Out[63]:

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

5 rows × 35 columns

In [64]:
```
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 [72]:
```
cols = ['BusinessTravel', 'Department', 'EducationField', 'Gender',
       'JobRole', 'MaritalStatus', 'Over18', 'OverTime']
```

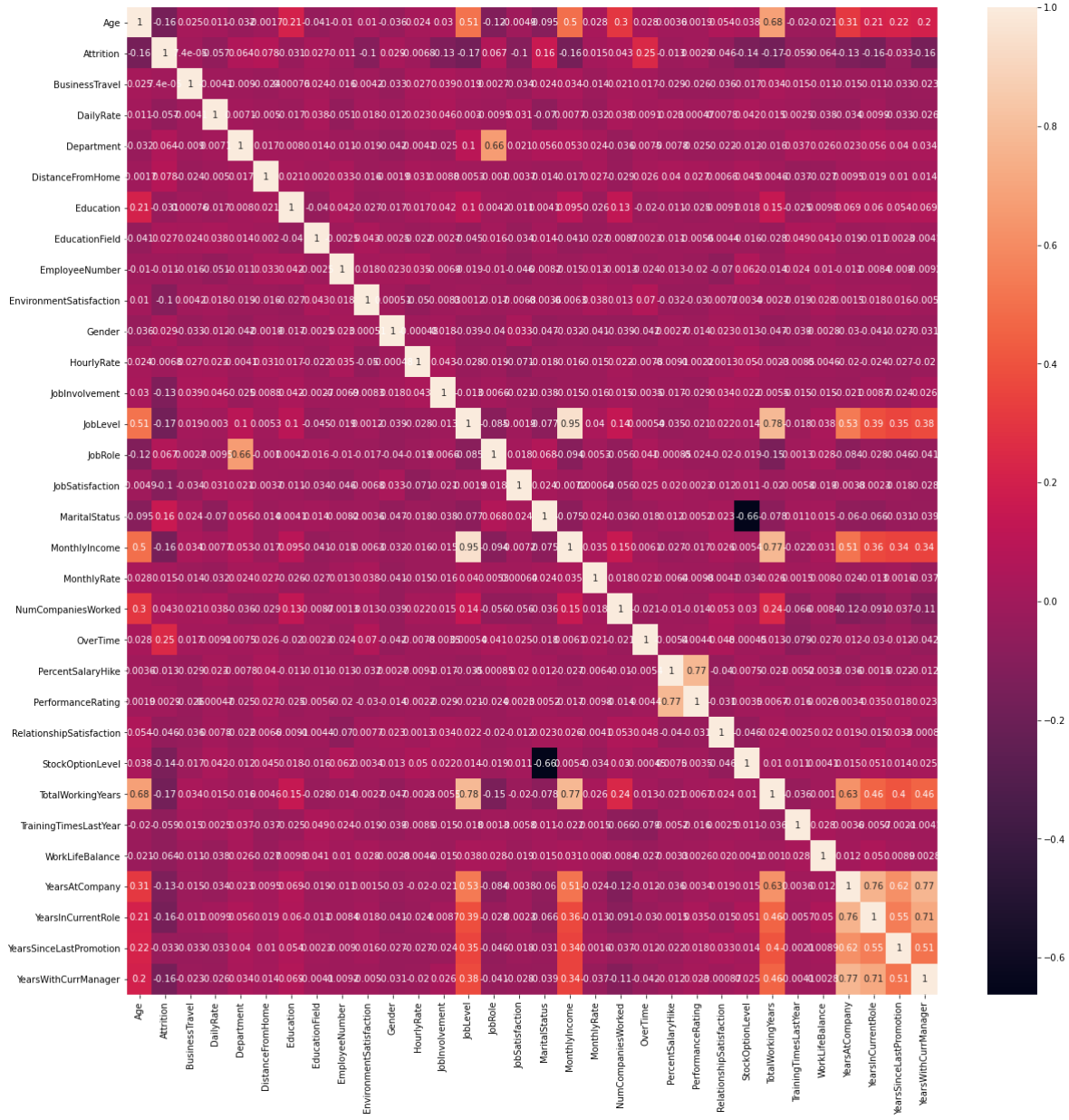
In [57]:
```
x = df.drop(['Attrition'],axis = 1)
```

In [58]:
```
y = df.Attrition
```

In [114]:
```
hp = df
```

In [145]:
```
plt.figure(figsize=(20,20))
sns.heatmap(hp.corr(),annot = True)
```

Out[145]: <AxesSubplot:>



In [153]:
```
print("Top 3 contributors to Attrition:\n",co.sort_values().iloc[-3:])
```

```
Top 3 contributors to Attrition:
 DistanceFromHome    0.077924
MaritalStatus        0.162070
OverTime             0.246118
Name: Attrition, dtype: float64
```

In [60]:
```
from sklearn.preprocessing import LabelEncoder
```

In [61]:
```
le = LabelEncoder()
```

In [73]:
```
for col in cols:
    x[col] = le.fit_transform(x[col])
```

In [74]:
```
x
```

Out[74]:

	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationFiel
0	41	2	1102	2	1	2	
1	49	1	279	1	8	1	
2	37	2	1373	1	2	2	
3	33	1	1392	1	3	4	
4	27	2	591	1	2	1	
...	...	...	...	...	...	...	
1465	36	1	884	1	23	2	
1466	39	2	613	1	6	1	
1467	27	2	155	1	4	3	
1468	49	1	1023	2	2	3	
1469	34	2	628	1	8	3	

1470 rows × 34 columns

In [75]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [76]:
```
ms = MinMaxScaler()
```

In [78]:
```
x = ms.fit_transform(x)
```

In [84]:
```
y = le.fit_transform(df.Attrition)
```

In [89]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

In [90]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_stat
```

In [96]:
```
lr = LogisticRegression()
tree = DecisionTreeClassifier()
```

In [97]:
```
lr.fit(x_train,y_train)
tree.fit(x_train,y_train)
```

Out[97]:
```
DecisionTreeClassifier()
```

In [98]:
```
log_pred = lr.predict(x_test)
```

In [99]:
```
tree_pred = tree.predict(x_test)
```

In [100]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [101]:
```
print(classification_report(y_test,log_pred))
```

```
              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 [102]:
```
print(classification_report(y_test,tree_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.84      0.86       371
           1       0.31      0.39      0.34        70

    accuracy                           0.77       441
   macro avg       0.59      0.61      0.60       441
weighted avg       0.79      0.77      0.78       441
```

Logistic Regression works better than Decision Trees

In [103]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [104]:
```
rfc = RandomForestClassifier(n_estimators=600)
```

In [105]:
```
rfc.fit(x_train,y_train)
```

Out[105]:
```
RandomForestClassifier(n_estimators=600)
```

In [107]:
```
predictions = rfc.predict(x_test)
```

In [108]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.87      0.99      0.93       371
           1       0.87      0.19      0.31        70

    accuracy                           0.87       441
   macro avg       0.87      0.59      0.62       441
weighted avg       0.87      0.87      0.83       441
```

Random Forest accuracy is very similar to Logistic Regression

In [ ]: