In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans,AgglomerativeClustering
from scipy.stats import zscore
from sklearn.metrics import silhouette_score,classification_report
import pandas as pd

pd.options.display.max_columns=1000
```

In [2]:

```
import pandas as pd
import pycaret
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
df.head()
```

Out[2]:

	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	

In [3]:

```
newdf = df[(df.Attrition == "Yes")]
for i in range(4):
    df =df.append(newdf)

df["Attrition"].value_counts()
```

Out[3]:

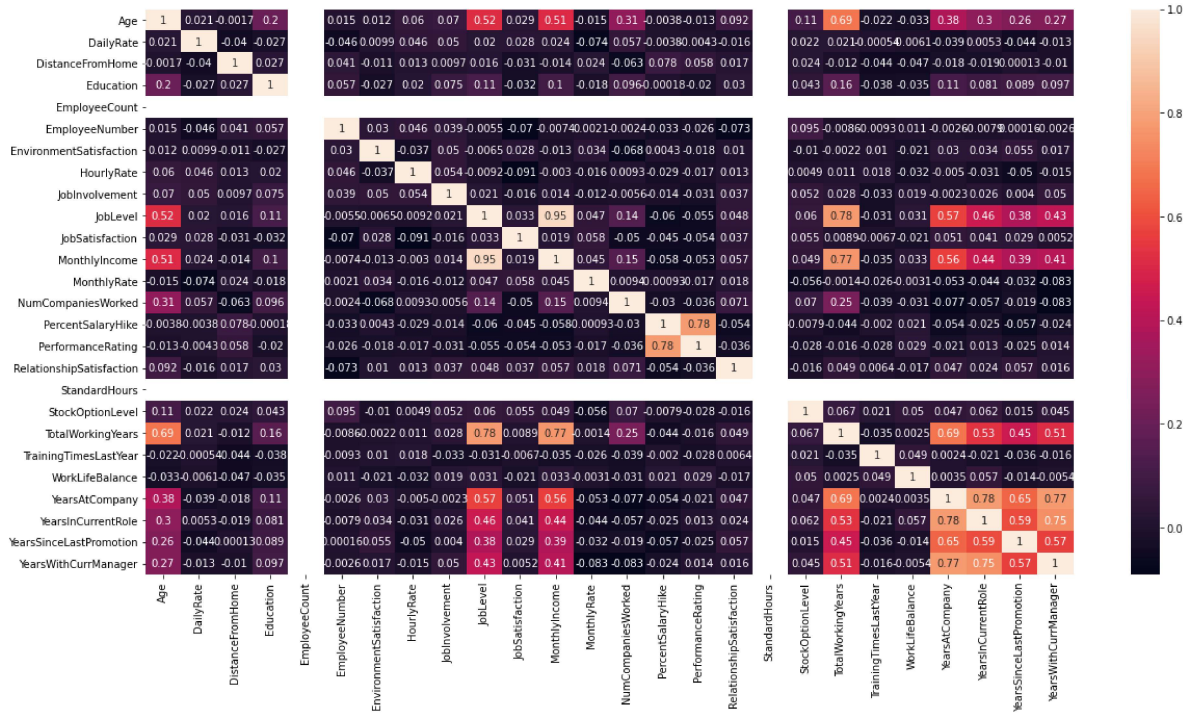
```
No     1233
Yes    1185
Name: Attrition, dtype: int64
```

In [4]:

```
#df.describe().transpose()
```

In [5]:

```
plt.figure(figsize=(20,10))
sns.heatmap(df.corr() , annot=True)
plt.show()
```



In [6]:

```
df.drop(columns=['EmployeeCount','StandardHours'] , inplace=True)
```

In [7]:

```
df_num=df.select_dtypes(exclude='object')
df_num_scaled=df_num.apply(zscore)

# encode categorical data
df_cat=df.select_dtypes(include='object')
df_cat_dummy=pd.get_dummies(df_cat, drop_first=True)


# concat numerical & categorical data
xscaled=pd.concat([df_num_scaled,df_cat_dummy] , axis=1).reset_index(drop=True)
xscaled.head()
```

Out[7]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfacti
0	0.566795	0.793394	-1.059524	-0.868959	-1.716348	-0.5495
1	1.410186	-1.247437	-0.212605	-1.851719	-1.714662	0.3357
2	0.145100	1.465405	-0.938535	-0.868959	-1.711291	1.2210
3	-0.276596	1.512520	-0.817547	1.096562	-1.709605	1.2210
4	-0.909139	-0.473757	-0.938535	-1.851719	-1.706233	-1.4348

In [8]:

```
xscaled["Attrition_Yes"].value_counts()
```

Out[8]:

```
0    1233
1    1185
Name: Attrition_Yes, dtype: int64
```

In [9]:

```
X = xscaled.drop(["Attrition_Yes"],axis=1)
```

In [10]:

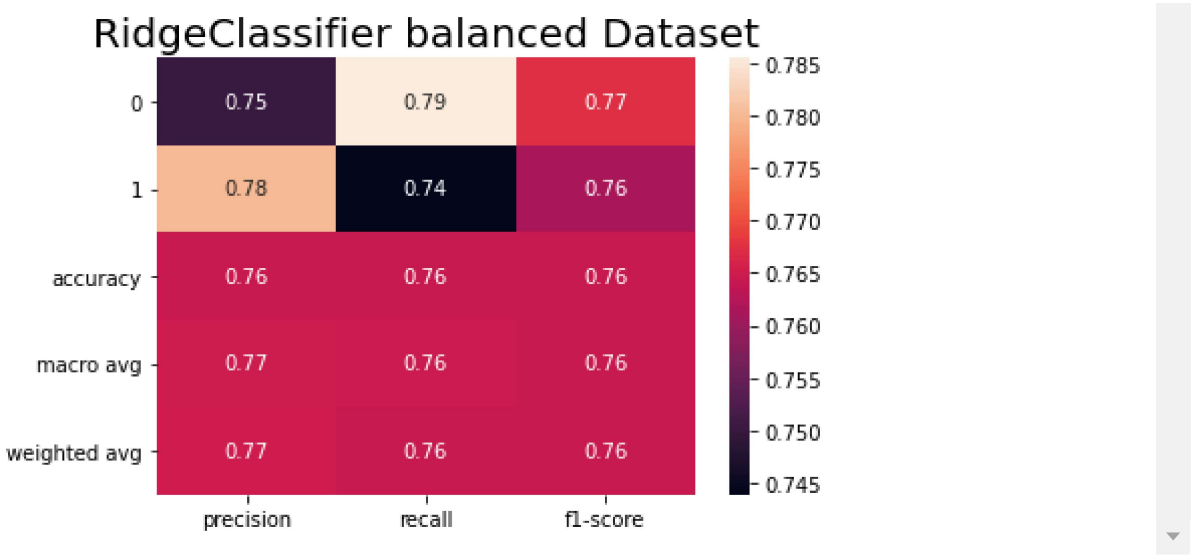
```
from sklearn.model_selection import train_test_split

y = xscaled["Attrition_Yes"]
```

RidgeClassifier balanced Dataset

In [11]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
from sklearn.linear_model import RidgeClassifier
rc = RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True,
                max_iter=None, normalize=False, random_state=4176,
                solver='auto', tol=0.001)
rc.fit(X_train,y_train)

y_pred=rc.predict(X_test)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
cf = classification_report(y_pred, y_test)
from pprint import pprint
print("\n")
pprint(cf)
from sklearn.metrics import classification_report, confusion_matrix
cm=  confusion_matrix(y_pred, y_test)
print("\n")
print(cm)

print("\n")
cf = classification_report(y_pred, y_test, output_dict=True)
P1 = sns.heatmap(pd.DataFrame(cf).iloc[:-1, :].T, annot=True).set_title("RidgeClassifier ba

plt.savefig("RidgeClassifier Imbalanced Dataset.jpg",bbox_inches="tight")
```

```
Accuracy: 0.7644628099173554


('              precision    recall  f1-score   support\n'
 '\n'
 '           0       0.75      0.79      0.77       359\n'
 '           1       0.78      0.74      0.76       367\n'
 '\n'
 '    accuracy                           0.76       726\n'
 '   macro avg       0.77      0.76      0.76       726\n'
 'weighted avg       0.77      0.76      0.76       726\n')


[[282  77]
 [ 94 273]]
```



In [12]:

```
from sklearn.model_selection import cross_val_score
f1 = cross_val_score(rc, X, y, scoring='f1', cv = 10)
print(f1)
```

```
[0.78861789 0.75949367 0.7804878  0.75949367 0.78838174 0.77310924
 0.76229508 0.74193548 0.75518672 0.75806452]
```

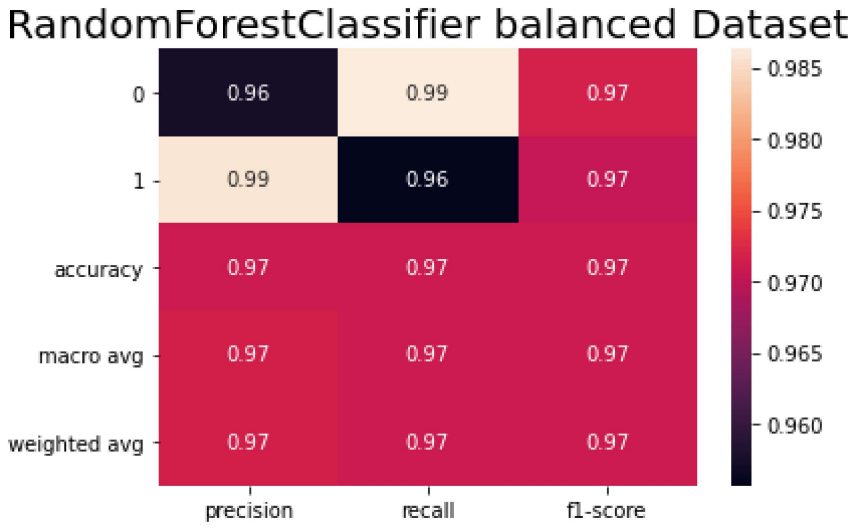
RandomForestClassifier balanced Dataset

In [13]:

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(
    n_estimators=100,
    criterion='gini',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features='auto',
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    min_impurity_split=None,
    bootstrap=True,
    oob_score=False,
    n_jobs=None,
    random_state=4176,
    verbose=0,
    warm_start=False,
    class_weight=None,
    ccp_alpha=0.0,
    max_samples=None,
)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

cf = classification_report(y_pred, y_test)
from pprint import pprint
print("\n")
pprint(cf)
from sklearn.metrics import classification_report, confusion_matrix
cm=  confusion_matrix(y_pred, y_test)
print("\n")
print(cm)

print("\n")
cf = classification_report(y_pred, y_test, output_dict=True)
P1 = sns.heatmap(pd.DataFrame(cf).iloc[:-1, :].T, annot=True).set_title("RandomForestClassi

plt.savefig("RandomForestClassifier Imbalanced Dataset.jpg",bbox_inches="tight")
```

```
Accuracy: 0.9710743801652892


('              precision    recall  f1-score   support\n'
 '\n'
 '           0       0.96      0.99      0.97       365\n'
 '           1       0.99      0.96      0.97       361\n'
 '\n'
 '    accuracy                           0.97       726\n'
 '   macro avg       0.97      0.97      0.97       726\n'
 'weighted avg       0.97      0.97      0.97       726\n')
```

```
[[360   5]
 [ 16 345]]
```



In [14]:

```
f1 = cross_val_score(clf, X, y, scoring='f1', cv = 10)
print(f1)
```

```
[0.87179487 0.98755187 0.98755187 0.99166667 0.97942387 0.97520661
 0.97520661 0.98333333 0.9874477  0.92913386]
```

In [15]:

```
print("F1 of Model with Cross Validation is:",f1.mean() * 100)
```

```
F1 of Model with Cross Validation is: 96.68317254700446
```