IBM HR Analytics Employee Attrition & Performance

Dataset Acquired from Kaggle. | **I don't own any rights on this dataset**

In this Assignment I'll be using Random Forest for easy and efficient model building. Since it's not that large dataset. It can be processed rather easily.

DATA SHAPE: 1,470 rows x 35 columns

IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

READING THE DATASET USING PANDAS LIBRARY

```
# Load the data
data = pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

DATA HEAD

In the given Dataset of IBM HR Analytics Employee Attrition & Performance, the initial approach should be checking the header of the data (including features name).

```
# Display the first 5 rows of the data
data.head()
print(data.head())
```

Using this, the dataset will be printed from the top till row 5 (including features name). In this way basic understanding of the data and the features name are to be identified in a look.

Here we can Assume 'Attrition' will be the target value that depends on other features/columns.

SEPERATING AND ANALYZING THE DATA

To further Analyze and clean the data, we need to separate Categorical Values [Object] and Numerical Values [int64].

```
# Seperating Categorical and Numerical Columns
cat_col = data.select_dtypes(include='object').columns
num_col = data.select_dtypes(exclude='object').columns
```

Here [object] data are separated in `cat_col` and [int64] data in `num_col` variables.

To Further Analyze the data, I can check Unique values in both Categorical and Numerical Data.

```
# Display the unique values of the Categorical Columns
for col in cat_col:
    print(col)
    print(data[col].unique())
```

```
# Display the unique values of the Numerical Columns
for col in num_col:
    print(col)
    print(data[col].unique())
```

CONVERTING CATEGORICAL DATA INTO NUMERICAL DATA USING LABEL ENCODING

The [object] or Categorical data cannot be further processed while it's not in Numerical Format. So here we will be using one of the technique **Label Encoding**, a technique in machine learning to convert Categorical Data into Numerical Data.

```
# converting categorical columns to numerical columns using Label Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in cat_col:
    data[col] = le.fit_transform(data[col])
```

again, to further check the encoded values by,

```
# Display the first 5 rows of the data
data.head()
print(data.head())
```

NOTE:

Since I'll be using Random Forest, further cleaning of data like removing outliers are not important in this model. Random Forest which uses several stacked Decision Tree model can work with this data.

CONCATINATION OF VARIABLES

The separated value of both Categorical and Numerical needs to be concatenated into one variable since we had already converted `cat_col` values into Numerical.

```
# concatenating the Categorical and Numerical Columns
new_data = pd.concat([data[cat_col], data[num_col]], axis=1)
print(new_data.head())
print(new_data.isna().sum())
```

After concatenation, I checked for any missing values alongside the head value.

SEPERATING TARGET AND MODEL BUILDING

Now we need to separate the Target [‘Attrition’] from the dataset and later split the whole data into training and test data for model building.

```
# Splitting the data into Features and Target
X = new_data.drop('Attrition', axis=1)
y = new_data['Attrition']

# Splitting the data into Training and Testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Training the model
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

EVALUATION OF THE DATASET

Prediction the accuracy of the built model, Confusion Matrix or Classification Report for depth evaluation.

```
# Evaluating the model
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Confusion Matrix:', confusion_matrix(y_test, y_pred))
print('Classification Report:', classification_report(y_test, y_pred))
```

RANDOM FOREST ACCURACY SCORE = 0.8843537414965986

NOTE: Check the Python ***.py*** file for further inquiries. CSV/Dataset is also present in this zip file.