HR ATTRITION

Submitted By - Namrata Sudhir Salve

HR Analytics -
Human Resource analytics deals with the analysis of the HR department in the organization. The goals of HR analytics are improving employee performance and getting better returns on investment.

HR Attrition -
Attrition in human resources refers to the gradual loss of employees overtime. Losing employees is not good for companies. Losing employees will cost more for the company because it will eventually involve hiring new employees and training them again. HR professionals have a responsibility to help the organization by maintaining a good work culture, good work environment, etc.

In this assignment, we need to predict whether a given employee will leave the organization or not.

We will create a model with the following steps:

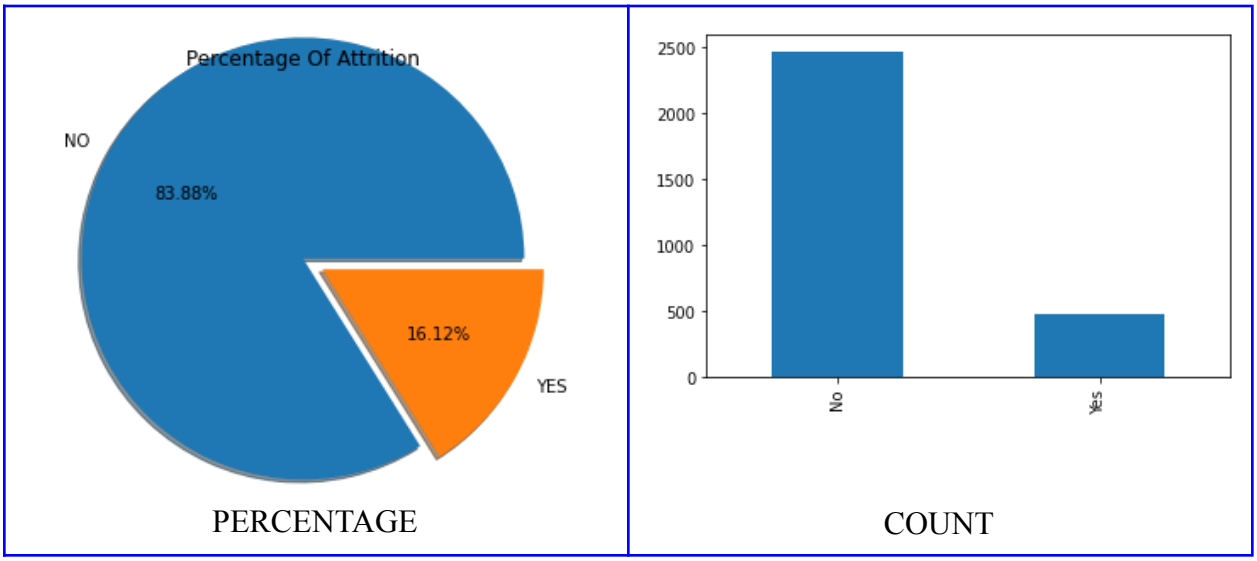
- Import the relevant packages
- Download and explore the dataset
- Perform EDA, Apply dataset for preprocessing
- Predict the target columns

What are the reasons or features on which attrition depends (features we have in the dataset) - Age, Business travel, Department, Distance from home, Education, Environment satisfaction, Gender, HourlyRate, Job Involvement, Job Level, Job Role, Job Satisfaction, Marital Status, Monthly Income, number of companies worked.

Data Analysis -

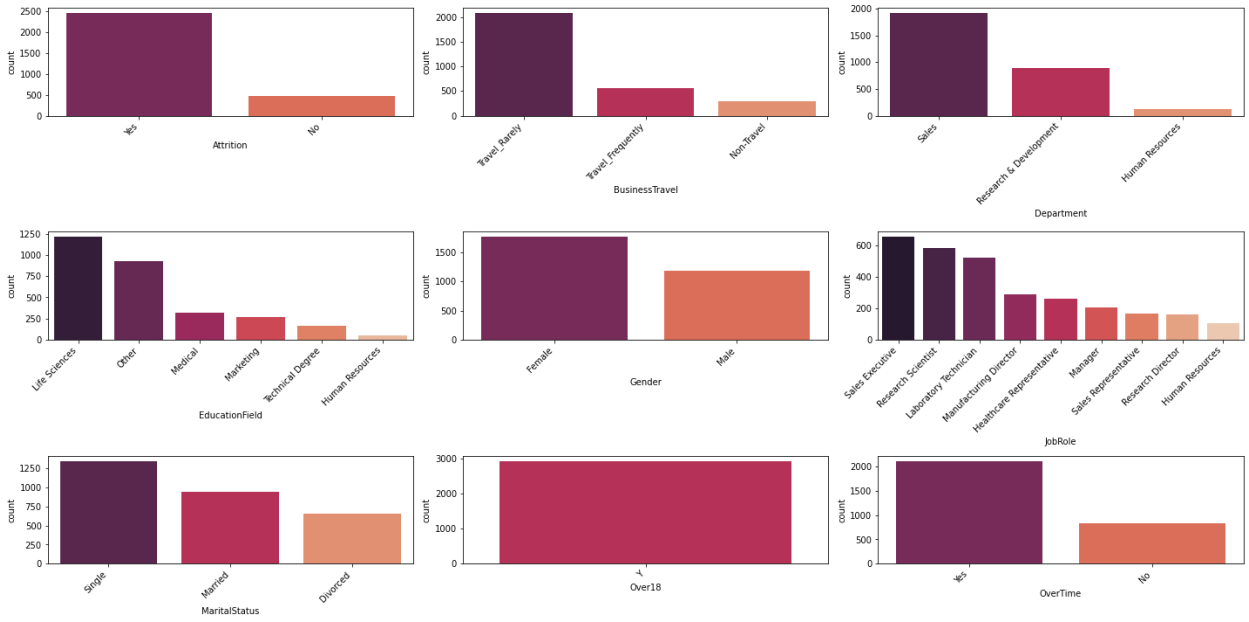
We will see the overall attrition percentage from the datasets.

ATTRITION

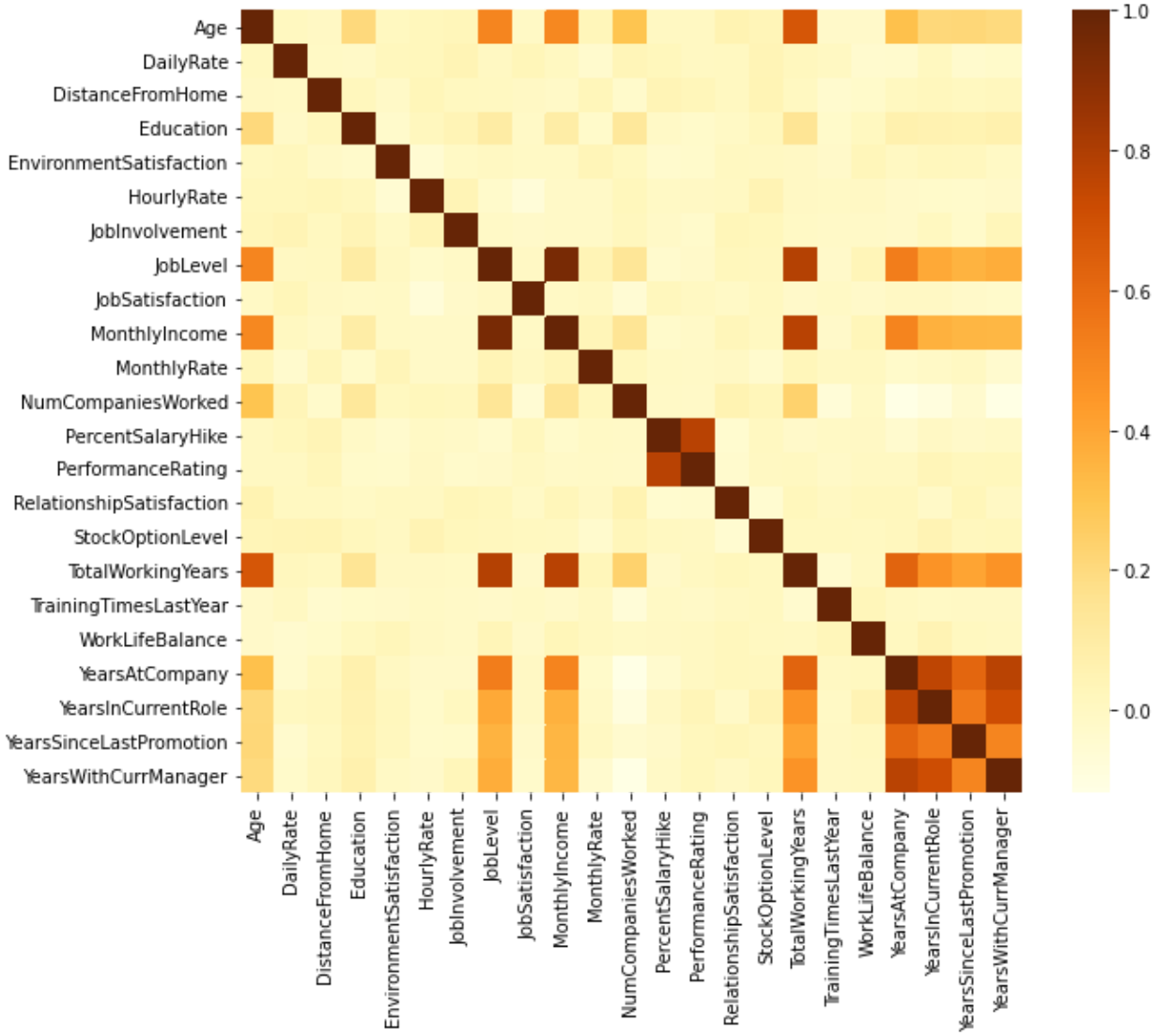


PERCENTAGE COUNT

Univariate Plots



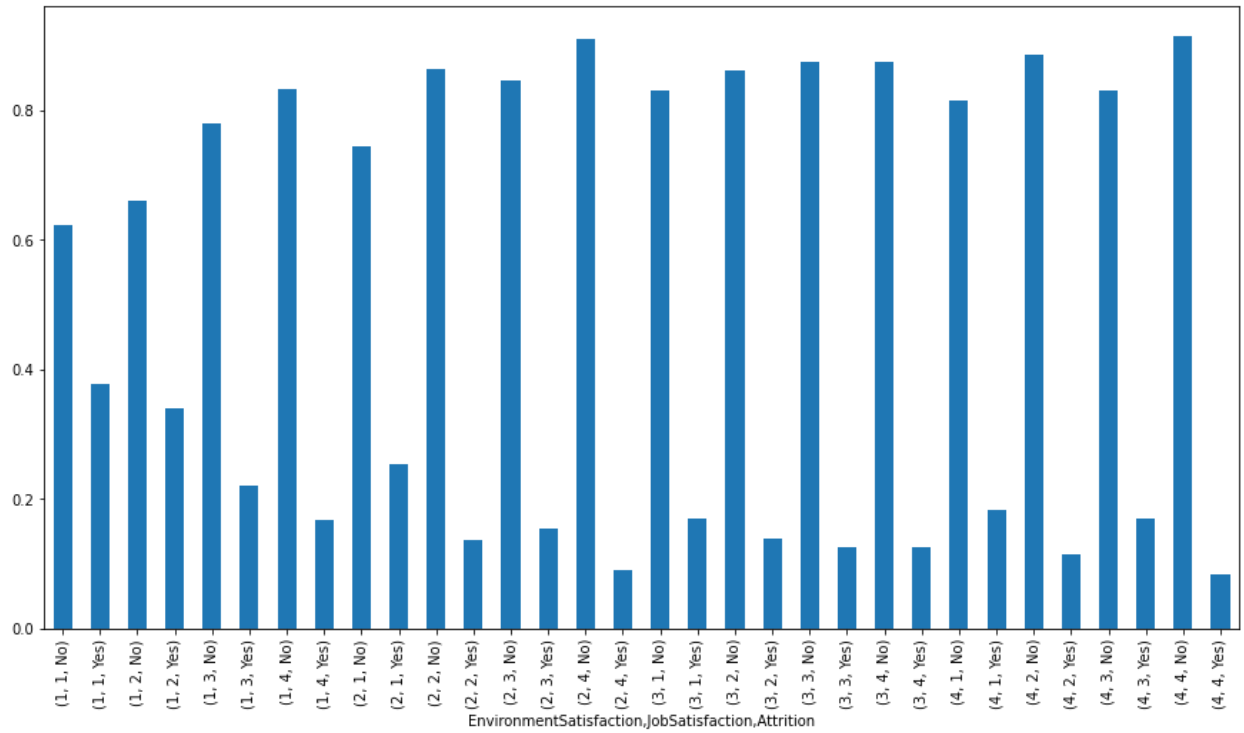
Plotting Correlations -



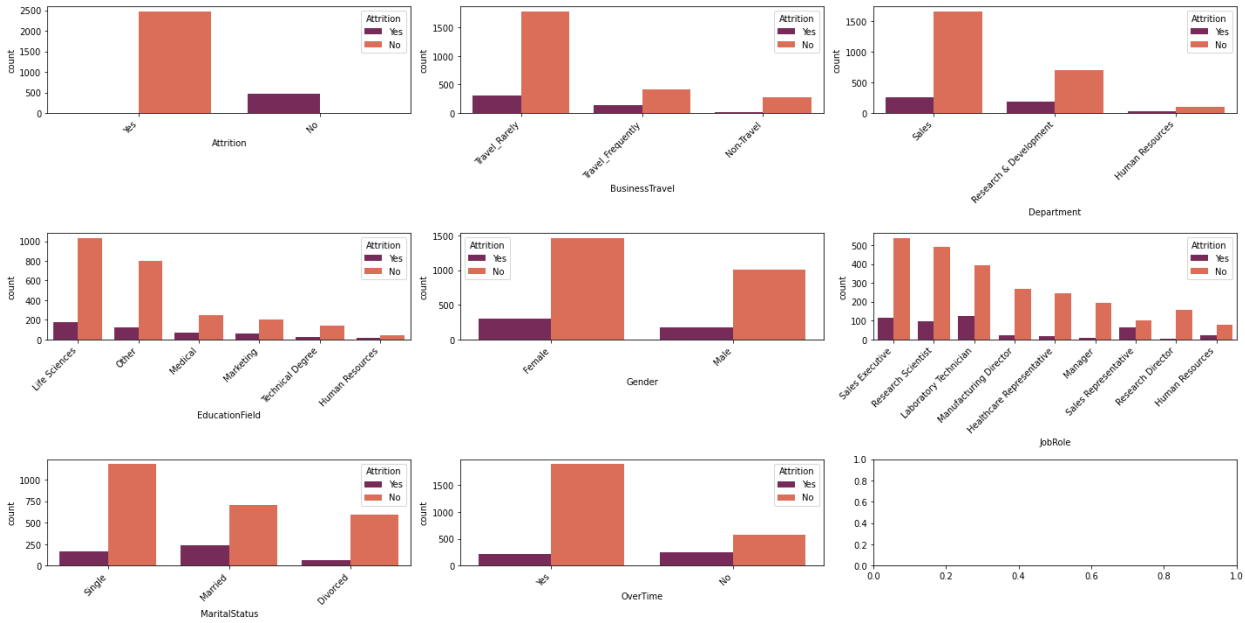
From the above matrix we can find features that affect attrition (target) features.
we can some features are related like -

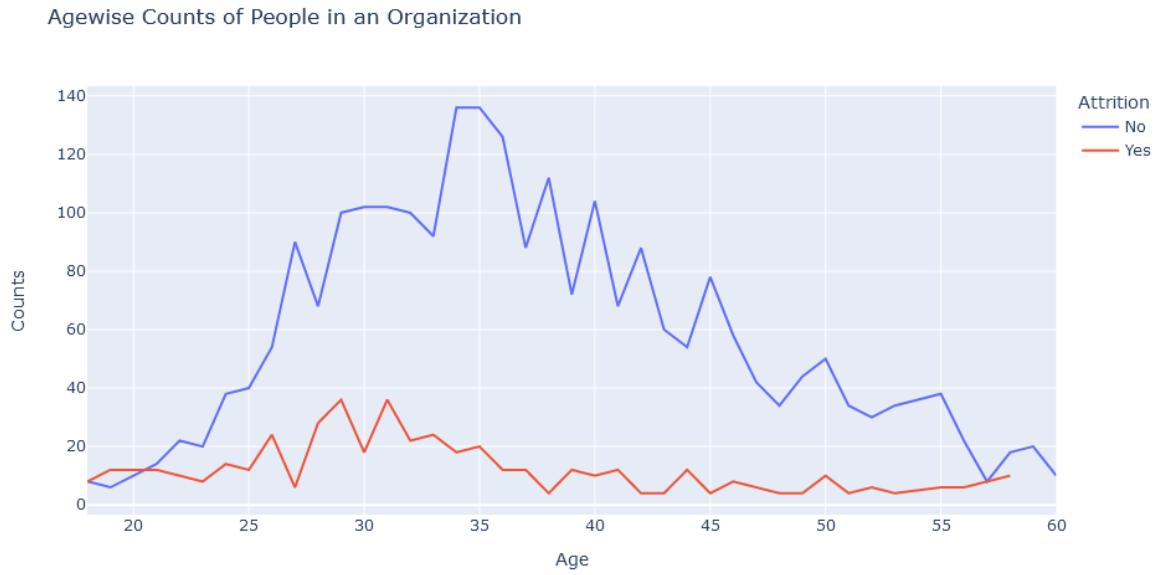
- Age and Total Working Years / Monthly Income / Job Level / Num of companies Worked
- Job Level and Total working years
- Job Level and Years at company
- Income and Working years
- Job Level and Monthly income

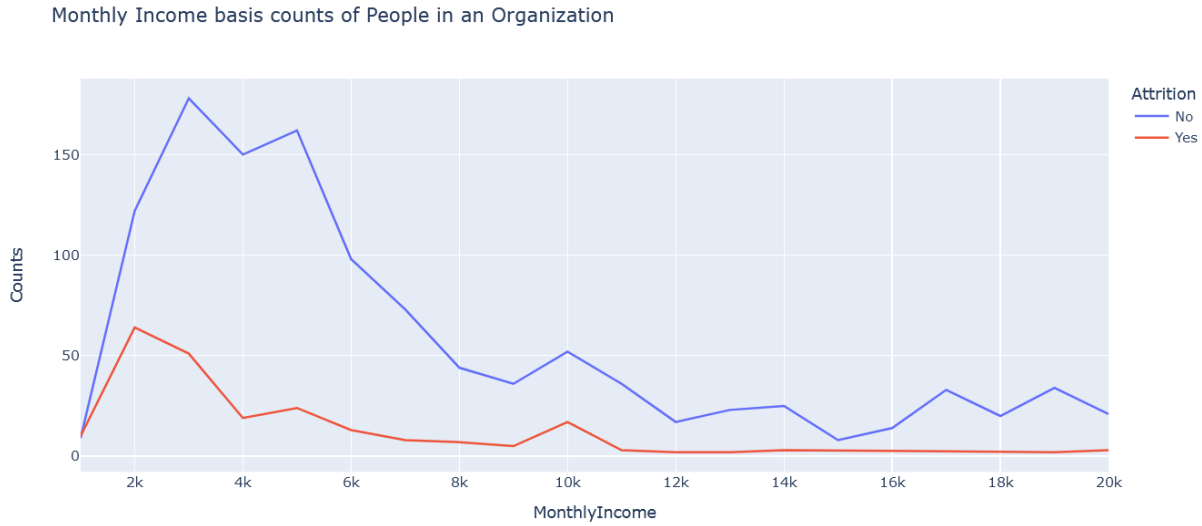
Attrition VS Environment Satisfaction and Job Satisfaction -



As we can see, attrition level decreases as environment satisfaction level and job satisfaction level increases.







As we can see, the attrition level is high at the initial level of ages (25 to 40) as age increases employee attrition level will decrease. The chances of older employees leaving the company are lesser.

Machine learning preprocessing -

```
def onehot_encode(df, column):
    df = df.copy()
    dummies = pd.get_dummies(df[column], prefix=column)
    df = pd.concat([df, dummies], axis=1)
    df = df.drop(column, axis=1)
    return df
```

```
def preprocess_inputs(df):
    df = df.copy()

    # Drop single-value columns and id columns
    df = df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18',
'StandardHours'], axis=1)

    # Binary-encode binary columns
    df['Gender'] = df['Gender'].replace({'Female': 0, 'Male': 1})
    df['OverTime'] = df['OverTime'].replace({'No': 0, 'Yes': 1})

    # Ordinal-encode the BusinessTravel column
    df['BusinessTravel'] = df['BusinessTravel'].replace({'Non-Travel': 0,
'Travel_Rarely': 1, 'Travel_Frequently': 2})

    # One-hot encoding
    for column in ['Department', 'EducationField', 'JobRole', 'MaritalStatus']:
        df = onehot_encode(df, column=column)

    # Split df into X and y
    y = df['Attrition']
    X = df.drop('Attrition', axis=1)

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
shuffle=True, random_state=1)

    # Scale X
    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train = pd.DataFrame(scaler.transform(X_train), index=X_train.index,
columns=X_train.columns)
    X_test = pd.DataFrame(scaler.transform(X_test), index=X_test.index,
```

```
columns=X_test.columns)

    return X_train, X_test, y_train, y_test
```

```
X_train, X_test, y_train, y_test = preprocess_inputs(data)
```

Splitting dataset for training and testing.

```
models = {
      "               Logistic Regression": LogisticRegression(),
      "               K-Nearest Neighbors": KNeighborsClassifier(),
      "                     Decision Tree": DecisionTreeClassifier(),
      "Support Vector Machine (Linear Kernel)": LinearSVC(),
      "   Support Vector Machine (RBF Kernel)": SVC(),
      "                     Random Forest": RandomForestClassifier(),
}

for name, model in models.items():
      model.fit(X_train, y_train)
      print(name + " trained.")
```

```
                  Logistic Regression trained.
                  K-Nearest Neighbors trained.
                        Decision Tree trained.
Support Vector Machine (Linear Kernel) trained.
   Support Vector Machine (RBF Kernel) trained.
                        Random Forest trained.
```

```
for name, model in models.items():
    print(name + ": {:.2f}%".format(model.score(X_test, y_test) * 100))
```

```
                  Logistic Regression: 88.10%
                  K-Nearest Neighbors: 85.37%
                        Decision Tree: 92.29%
Support Vector Machine (Linear Kernel): 87.98%
   Support Vector Machine (RBF Kernel): 89.46%
                        Random Forest: 94.78%
```

For classification I have compared classification algorithms such as Logistic regression, KNN,

Decision tree, SVM, Random forest.

Random forest and **decision tree** gives the highest accuracy of **94.78%** and **32.29%** respectively.