* **Predict retention of an employee within an organization such that whether the employee will leave the company or continue with it. An organization is only as good as its employees, and these people are the true source of its competitive advantage. Dataset is HR\_comma\_sep.csv and first do data exploration and visualization, after this create a logistic regression model to predict Employee Attrition.**
* import numpy as np
* import pandas as pd
* import matplotlib.pyplot as plt
* import seaborn as sns
* import warnings
* warnings.filterwarnings("ignore")
* hr = pd.read\_csv("HR\_comma\_sep.csv")
* hr
* hr.info()
* hr.isnull().sum()
* hr.describe()
* hr.head()
* hr["left"].value\_counts()
* hr["Department"].value\_counts().plot(kind = "pie", autopct = "%1.1f%%")
* plt.show()
* plt.hist(hr["time\_spend\_company"])
* plt.show()
* plt.scatter(x = hr["satisfaction\_level"], y = hr["salary"])
* plt.show()
* sns.boxplot(data = hr, x = "satisfaction\_level", y = "salary")
* plt.show()
* sns.scatterplot(x = hr["satisfaction\_level"], y = hr["salary"])
* plt.show()
* hr\_num = hr.select\_dtypes(["int","float"])
* hr\_num
* hr\_cat = hr.select\_dtypes(["object"])
* hr\_cat
* from sklearn.preprocessing import LabelEncoder
* le = LabelEncoder()
* for col in hr\_cat:
* hr\_cat[col] = le.fit\_transform(hr\_cat[col])
* hr\_cat
* le.classes\_
* hr\_new = pd.concat([hr\_num, hr\_cat], axis = 1)
* hr\_new
* x = hr\_new.drop("left", axis = 1)
* x
* y = hr\_new.iloc[:,-4]
* y
* from sklearn.model\_selection import train\_test\_split
* xtrain, xtest, ytrain, ytest = train\_test\_split(x,y, test\_size = 0.3, random\_state = 1)
* from sklearn.linear\_model import LogisticRegression
* lr = LogisticRegression()
* lr.fit(xtrain, ytrain)
* ypred = lr.predict(xtest)
* from sklearn.metrics import classification\_report
* print(classification\_report(ytest,ypred))
* lr = LogisticRegression(solver = "saga")
* lr.fit(xtrain, ytrain)
* ypred = lr.predict(xtest)
* print(classification\_report(ytest,ypred))
* lr = LogisticRegression(solver = "liblinear")
* lr.fit(xtrain, ytrain)
* ypred = lr.predict(xtest)
* print(classification\_report(ytest,ypred))