**DATA TRAINED ACADEMY**

**COURSE :- PG PROGRAME IN DATA SCIENCE**

**Article Title:***Predicting Employee Attrition and Performance*

*through HR Analytics*

**ARTICLE WRITTEN BY** ASHISH KUMAR PATEL

**COURSE NAME:-**  PG PROGRAME IN DATA SCIENCE

**SME NAME:**Mohd. KASHIF SIR

**Date of Submission:**3rd August 2024

**HR Analytics Employee Attrition and Performance Prediction**

**----------------------------------------------------------------------------------**

**1.Problem Definition**

Employee attrition is a major concern for organizations, as it can lead to significant costs and disruptions to business operations. The HR Analytics Employee Attrition and Performance dataset provides a unique opportunity to explore the factors contributing to employee attrition and develop predictive models to identify at-risk employees.

**. Problem Statement**

To build the machine learning model and predict employee “Attrition” based on various HR analytics metrics.

**. Import the Library**

|  |
| --- |
| # import libraries  import pandas as pd  import numpy as np  import seaborn as sns  import matplotlib.pyplot as plt  import warnings  warnings.filterwarnings('ignore') |

**. Read The Data**

|  |
| --- |
| data = pd.read\_csv("ibm-hr-analytics-employee-attrition-performance.zip") |
| data.head |

**2. Data Analysis**

This dataset contains 1470 rows and 35 columns in the data and the details of HR Analytics case study in this dataset numerical and categorical data is present. Here Attrition is our target/output veriable, Attrition which contains categorical so it will be termed as a classification problem where we need to predict the several Attrition using the classification problem.

**Here Categorical and Numerical columns**

**Categorical Variables:**

* Business Travel
* Department
* Education Field
* Gender
* Job Role
* Marital Status
* Over18
* Over Time

**Numerical Variables:**

* Age
* Daily Rate
* Distance From Home
* Education
* Environment Satisfaction
* Hourly Rate
* Job Involvement
* Job Level
* Job Satisfaction
* Monthly Income
* Monthly Rate
* Num Companies Worked
* Percent Salary Hike
* Performance Rating
* Stock Option Level
* Total Working Years
* Training Times Last Year
* Work Life Balance
* Years At Company
* Years In Current Role
* Years Since Last Promotion
* Years With Curr Manager

**Target variable: -**

* Attrition

**Data Summary**

Here’s an example of initial findings you might encounter:

* **Numerical Features**: The average Monthly Income might be $5,000, while the average Distance From Home is 10 miles.
* **Categorical Features**: Job Role might include roles like Sales, Executive, Research Scientist and Department could be Sales, Human Resources & Research & Development etc.

**3. EDA** (**Concluding Remarks**)

InExploratory Data Analysis we check the data and shape of the datatype so 1470 rows and 35 columns present in this data and int64 and object datatype present in the data and similarly we check the unique value of the data and we see in this dataset different different columns different value are present then we check the null value of the data so in this data no null value are present then check the information about the data so we find there are int64(26), object(9) values are present .

|  |
| --- |
| 1# check the columns of dataset  data.columns.tolist() |
| 2# check the data types of data  data.dtypes |
| 3 # check the null value of the data  data.isnull().sum() |
| 4 # check the information about the data  data.info() |

|  |
| --- |
| # check the statistical information  data.describe() |

Here check the statistical information of numerical columns. In the statistical dataset we check the count of the data then mean of the data then standard deviation of the data then min value of the data which is also called a q1 of the data then 25% , 50%, 75%, and max of the dataset so we see the max dataset of some columns are high like, Total Working Years and Years At Company, Years In Current Role, Years Since Last Promotion, Years With Curr Manager and other columns data looks perfect since there is no negative/invalid values present The counts of all the columns are same which means in this dataset no missing values in the dataset.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| **Age** | 1470.000000 | 36.923810 | 9.135373 | 18.000000 | 30.000000 | 36.000000 | 43.000000 | 60.000000 |
| **Daily Rate** | 1470.000000 | 802.485714 | 403.50910 | 102.000000 | 465.000000 | 802.000000 | 1157.000000 | 1499.000000 |
| **Distance**  **From Home** | 1470.000000 | 9.192517 | 8.106864 | 1.000000 | 2.000000 | 7.000000 | 14.000000 | 29.000000 |
| **Education** | 1470.000000 | 2.912925 | 1.024165 | 1.000000 | 2.000000 | 3.000000 | 4.000000 | 5.000000 |
| **Employee**  **Count** | 1470.0 | 1.0 | 0.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| **Employee**  **Number** | 1470.000000 | 1024.865306 | 602.024335 | 1.000000 | 491.250000 | 1020.500000 | 1555.750000 | 2068.000000 |
| **Environment**  **Satisfaction** | 1470.000000 | 2.721769 | 1.093082 | 1.000000 | 2.000000 | 3.000000 | 4.000000 | 4.000000 |
| **Hourly Rate** | 1470.000000 | 65.891156 | 20.329428 | 30.000000 | 48.000000 | 66.000000 | 83.750000 | 100.000000 |
| **Job Involvement** | 1470.000000 | 2.729932 | 0.711561 | 1.000000 | 2.000000 | 3.000000 | 3.000000 | 4.000000 |
| **Job Level** | 1470.000000 | 2.063946 | 1.106940 | 1.000000 | 1.000000 | 2.000000 | 3.000000 | 5.000000 |
| **Relationship**  **Satisfaction** | 1470.000000 | 2.712245 | 1.081209 | 1.000000 | 2.000000 | 3.000000 | 4.000000 | 4.000000 |
| **Standard Hours** | 1470.0 | 80.0 | 0.0 | 80.0 | 80.0 | 80.0 | 80.0 | 80.0 |
| **Stock Option**  **Level** | 1470.000000 | 0.793878 | 0.852077 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 3.000000 |
| **Total Working**  **Years** | 1470.000000 | 11.279592 | 7.780782 | 0.000000 | 6.000000 | 10.000000 | 15.000000 | 40.000000 |
| **Training Times**  **Last Year** | 1470.0000000 | 2.799320 | 1.289271 | 0.000000 | 2.000000 | 3.000000 | 3.000000 | 6.000000 |
| **Work Life**  **Balance** | 1470.000000 | 2.761224 | 0.706476 | 1.000000 | 2.000000 | 3.000000 | 3.000000 | 4.000000 |
| **Years At Company** | 1470.000000 | 7.008163 | 6.126525 | 0.000000 | 3.000000 | 5.000000 | 9.000000 | 40.000000 |
| **Years In Current Role** | 1470.000000 | 4.229252 | 3.623137 | 0.000000 | 2.000000 | 3.000000 | 7.000000 | 18.000000 |
| **Years Since Last**  **Promotion** | 1470.000000 | 2.187755 | 3.222430 | 0.000000 | 0.000000 | 1.000000 | 3.000000 | 15.000000 |
| **Years With Curr Manager** | 1470.000000 | 4.123129 | 3.568136 | 0.000000 | 2.000000 | 3.000000 | 7.000000 | 17.000000 |

**Visualization Techniques**

1. **Countplot**: count plot use to visualize the shape of the Target variable which is categorical features like Attrition, and more we visualize the data using count plot like, Business Travel, Department, Education Field, Gender, Job role, Married status, Over time This helps in understanding the range and shape of the data.

|  |
| --- |
| # check the attrition  sns.countplot(x=data.Attrition)  plt.show() |

1. **Histplot** :- hist plot use to visualize the int type of data using target variable and here we use to columns like Age ,Daily Rate, Distance From Home, Employee Number, Hourly Rate, Monthly Income, Monthly Rate, Num Companies Worked, Percent Salary Hike, Total Working Years, Years At Company, Years In Current Role, Years Since Last Promotion, Years With Curr Manager.

|  |
| --- |
| # visualize the data using histplot  plt.figure(figsize=(25,22), facecolor='white')  plotnumber = 1  for column in data1:  if plotnumber<=20:  ax = plt.subplot(5,4,plotnumber)  sns.histplot(x=data1[column].dropna(axis=0),hue=data.Attrition)  plt.xlabel(column, fontsize=20)  plt.ylabel('Attrition', fontsize=20)  plotnumber+=1  plt.show() |

1. **Distplot** :- here distribution plot use to visualize the data and check the skewness of the data here we use the columns Daily Rate, Distance From Home, Employee Number, Hourly Rate, Monthly Income, Monthly Rate, Num Companies Worked, Percent Salary Hike, Total Working Years, Years At Company, Years In Current Role, Years Since Last Promotion Years With Curr Manager.

|  |
| --- |
| # visualize the data using distplot  plt.figure(figsize=(28,22), facecolor='white')  plotnumber = 1  for column in data1:  if plotnumber<=20:  ax = plt.subplot(5,4,plotnumber)  sns.distplot(x=data1[column].dropna(axis=0))  plt.xlabel(column, fontsize=20)  plt.ylabel('Attrition', fontsize=20)  plotnumber+=1  plt.show() |

1. **Box Plots**: Use box plots to identify outliers in features such as Age, daily Rate, Distance From Home, Education, Environment Satisfaction, Hourly Rate, Job Involvement, Job Level, Job Satisfaction, Monthly Income, Performance Rating, Relationship Satisfaction, Stock Option Level, Total Working Years, Training Times Last Year, Work Life Balance, Years At Company, Years In Current Role, Years Since Last Promotion, Years With Curr Manager Box plots provide insights into the data’s spread and variability.

|  |
| --- |
| # check the skewness using boxplot  plt.figure(figsize=(20,30), facecolor='white')  plotnumber = 1  for col in data:  if plotnumber<=32:  ax=plt.subplot(8,4,plotnumber)  sns.boxplot(data[col], color="m")  plt.xlabel(col, fontsize=10)  plotnumber+=1  plt.show() |

1. **Correlation Matrix**: Generate a correlation matrix to explore relationships between numerical features and the target variable Left. Use a heatmap to visualize correlations and identify features that are strongly correlated with turnover.

|  |
| --- |
| # check the correlation using heatmap  plt.figure(figsize=(25,25))  sns.heatmap(data.corr(),annot=True)  plt.show() |

**Key Insights**

From the exploratory data analysis (EDA), several insights emerge:

* **Significant Correlations:** Job Satisfaction and Distance From Home show significant correlations with attrition. Employees with lower job satisfaction and longer commutes are more likely to leave.
* **Feature Insights:** The distribution of features like Monthly Income and Age indicates variability in employee profiles, which can be relevant for predicting attrition.

**4. Pre-processing Pipeline**

**1 Data Cleaning**

Data cleaning involves handling missing values and outliers. For this dataset, we check for and address any missing values.

|  |
| --- |
| # Check for missing values  print(data.isnull().sum())  # Example: Drop rows with missing values (if any)  data = data.dropna() |

**2 Encoding Categorical Variables**

Categorical variables need to be converted into numerical formats using techniques such as Ordinal Encoder to be used in machine learning models.

|  |
| --- |
| # OrdinalEncoder categorical features  from sklearn.preprocessing import OrdinalEncoder  ord\_enc = OrdinalEncoder()  for i in data.columns:  if data [i].dtypes=='object':  data[i] = ord\_enc.fit\_transform(data[i].values.reshape(-1,1))  data |

**3. Split the features and target variable**

|  |
| --- |
| # split the features and target  X = data.drop("Attrition",axis=1)  Y = data["Attrition"] |

**4.Feature Scaling**

Normalization or scaling of numerical features ensures that all features contribute equally to the model's performance using Standard Scaler.

|  |
| --- |
| # scaling the data  from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  X = pd.DataFrame(scaler.fit\_transform(X),columns = X.columns)  X |

**5 Check the variance**

Check the outliers using variance inflation factor (VIF) and remove the outliers .

|  |
| --- |
| # Check and remove the outliers  vif = pd.DataFrame()  vif['VIF values'] = [variance\_inflation\_factor(X.values, i ) for i in range(len(X.columns))]  vif['Features'] = X.columns  vif |

**6 Resample the data**

Resample the data using SMOTE function

|  |
| --- |
| # Resample the data using smote  from imblearn.over\_sampling import SMOTE  SM = SMOTE()  X1, Y1= SM.fit\_resample(X,Y) |

**7 Feature Selection**

Select relevant features for modeling based on correlation analysis and domain knowledge. This step helps in reducing dimensionality and improving model performance.

1. **Building Machine Learning Models**

**1 Find the best accuracy using Logistic Regression**

|  |
| --- |
| # Find the best accuracy  maxAccu = 0  maxRS = 0  for i in range (1,200):  x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,Y, test\_size = .30, random\_state=i)  RFR = RandomForestClassifier()  RFR.fit(x\_train, y\_train)  pred = RFR.predict(x\_test)  acc = accuracy\_score(y\_test, pred)  if acc>maxAccu:  maxAccu = acc  maxRS = i  print("Best Accuracy is ", maxAccu, "at random\_state", maxRS) |

**2 Train and test the data using train\_test\_split**

|  |
| --- |
| # Train and test the data  x\_train, x\_test, y\_train, y\_test = train\_test\_split(X,Y, test\_size = .30, random\_state = maxRS) |

**3 Model Selection for different different models:-**

We choose several machine learning models to predict employee attrition:

* **Logistic Regression:** A baseline model for binary classification.

|  |
| --- |
| # Logistic Regression  LR = LogisticRegression()  LR.fit(x\_train, y\_train)  pred = LR.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **Random Forest Classifier:** A robust model that handles complex relationships.

|  |
| --- |
| #Random Forest Classifier  RFC = RandomForestClassifier()  RFC.fit(x\_train, y\_train)  pred = RFC.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **Support Vector Classifier:** The support vector is use to find the best score.

|  |
| --- |
| # Support Vector Classifier  svc = SVC()  svc.fit(x\_train, y\_train)  pred = svc.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **Gradient Boosting Classifier (GBC):** An advanced model for better accuracy.

|  |
| --- |
| # Gradient Boosting Classifier  GBC = GradientBoostingClassifier()  GBC.fit(x\_train, y\_train)  pred = GBC.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **Ada Boost Classifier:** This model is also advance for find the better accuracy.

|  |
| --- |
| # AdaBoostClassifier  ada = AdaBoostClassifier()  ada.fit(x\_train, y\_train)  pred = ada.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **Bagging Classifier:** Use this model to taking random subsets of an original dataset to find the best accuracy.

|  |
| --- |
| # BaggingClassifier  Bagg = BaggingClassifier()  Bagg.fit(x\_train, y\_train)  pred = Bagg.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **Decision Trees Classifier :** This model take to It segments data based on features to make decisions and predict outcomes for better accuracy.

|  |
| --- |
| # DecisionTreeClassifier  DT = DecisionTreeClassifier()  DT.fit(x\_train, y\_train)  pred = DT.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **Extra Trees Classifier:** Use this model to improve the predictive accuracy and control over-fitting the data and find the better accurcay.

|  |
| --- |
| # ExtraTreesClassifier  ETC = ExtraTreesClassifier()  ETC.fit(x\_train, y\_train)  pred = ETC.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

* **KNN:** valuable for tackling classification challenges, particularly in scenarios with non-linear or complex decision boundaries.

|  |
| --- |
| # KNN  knn = KNN()  knn.fit(x\_train, y\_train)  pred = knn.predict(x\_test)  print(accuracy\_score(y\_test, pred))  print(confusion\_matrix(y\_test, pred))  print(classification\_report(y\_test, pred)) |

1. **Check the cross validation score for all the models :-**

* **Logistic Regression:** A baseline model for binary classification for checking the cross validation.

|  |
| --- |
| # cross validation Logistic Regression  score = cross\_val\_score(LR, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **Random Forest Classifier :** A robust model that handles complex relationships for checking the cross validation.

|  |
| --- |
| # cross validationRandom Forest Classifier  score = cross\_val\_score(RFC, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **Support Vector Classifier:** The support vector is use to find the best score for checking the cross validation.

|  |
| --- |
| # cross validation Support Vector Classifier  score = cross\_val\_score(svc, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **Gradient Boosting Classifier (GBC):** An advanced model for better accuracy for checking the cross validation.

|  |
| --- |
| # cross validation Gradient Boosting classifier  score = cross\_val\_score(GBC, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **Ada Boost Classifier:** This model is also advance for find the better accuracy for checking the cross validation.

|  |
| --- |
| # cross validation Ada Boost Classifier  score = cross\_val\_score(ada, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **Bagging Classifier:** Use this model to taking random subsets of an original dataset to find the best accuracy for checking the cross validtion.

|  |
| --- |
| # cross validation Bagging Classifier  score = cross\_val\_score(Bagg, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **Decision Trees Classifier :** This model take to It segments data based on features to make decisions and predict outcomes for better accuracy for checking the cross validation.

|  |
| --- |
| # cross validation Decision Trees Classifier  score = cross\_val\_score(DT, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **Extra Trees Classifier:** Use this model to improve the predictive accuracy and control over-fitting the data and find the better accuracy for checking the cross validation.

|  |
| --- |
| # cross validation Extra Trees Classifier  score = cross\_val\_score(ETC, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

* **KNN:** valuable for tackling classification challenges, particularly in scenarios with non-linear or complex decision boundaries for checking the cross validation.

|  |
| --- |
| # Cross validation KNN  score = cross\_val\_score(knn, X1, Y1)  print(score)  print(score.mean())  print("Difference between accuracy score and cross validation score is -",accuracy\_score(y\_test, pred)- score.mean()) |

1. **Hyper parameter tuning for Best Model:-** Based on model evaluation metrics, we select the best-performing model for final predictions. The Extra Trees Classifier often yields superior results due to its ability to capture complex interactions between features using GridSearchCV.

|  |
| --- |
| #Hyperparameter tuning for best model  from sklearn.model\_selection import GridSearchCV  parameters = {'criterion': ['gini', 'entropy'],  'random\_state':[10,50,300,800,1000],  'max\_depth':[0,10,20],  'n\_jobs':[-2,-1,1],  'n\_estimators':[50,100,150,200]}  GSCV = GridSearchCV(ExtraTreesClassifier(), parameters, cv = 5)  GSCV.fit(x\_train, y\_train)  GSCV.best\_params\_  Best\_model = ExtraTreesClassifier(criterion= 'gini', max\_depth= 10, n\_estimators= 200, n\_jobs= -2, random\_state= 1000)  Best\_model.fit(x\_train, y\_train)  pred = Best\_model.predict(x\_test)  acc = accuracy\_score(y\_test, pred)  print(acc\*100) |

1. **Save the best model :-** Save the model which model tune the hyperparameter .

|  |
| --- |
| # save the best model  import joblib  joblib.dump(Best\_model,"HR\_analytics\_evalution.pkl") |

1. **Predict the saved model:-**

|  |
| --- |
| # Predict the saved model  model = np.array(y\_test)  data = pd.DataFrame()  data["predicted"] = prediction  data["original"] = model  data |

**6.Concluding Remarks**

**1 Summary of Findings**

Our machine learning models have provided valuable insights into employee attrition:

* **Model Performance:** The Extra Trees Classifier demonstrates high accuracy and robustness in predicting attrition, outperforming the Logistic Regression model.
* **Key Predictors:** Features such as job satisfaction and commute distance are crucial in predicting employee turnover.

**2 Practical Implications**

**For HR Departments:**

* **Retention Strategies:** Focus on improving job satisfaction and addressing long commutes to reduce attrition rates. Implementing these insights can help in developing targeted retention strategies.

**For Companies:**

* **Cost Efficiency:** Proactive measures based on predictive insights can lead to cost savings and better employee satisfaction, ultimately enhancing overall organizational performance.

**3 Limitations and Future Work**

**Limitations:**

* **Data Constraints:** The analysis is limited to the features available in the dataset.
* **Model Limitations:** Potential issues such as overfitting or biases in model predictions.

**Future Enhancements:**

* **Additional Data:** Incorporate external data sources to enrich the analysis.
* **Advanced Models:** Explore other algorithms or ensemble methods to improve predictive accuracy further.

**Conclusion**

By applying machine learning techniques to HR Analytics dataset, we have successfully developed models to predict employee attrition and gain valuable insights into factors influencing turnover. This analysis underscores the importance of data-driven decision-making in HR practices, highlighting how predictive analytics can aid in developing effective retention strategies. As organizations continue to face challenges related to employee retention, leveraging advanced analytics will become increasingly crucial in maintaining a competitive edge and fostering a productive and satisfied workforce.