SmartInternz

AIML – Assignment 4

Rahul Palanivel

21BCE7828

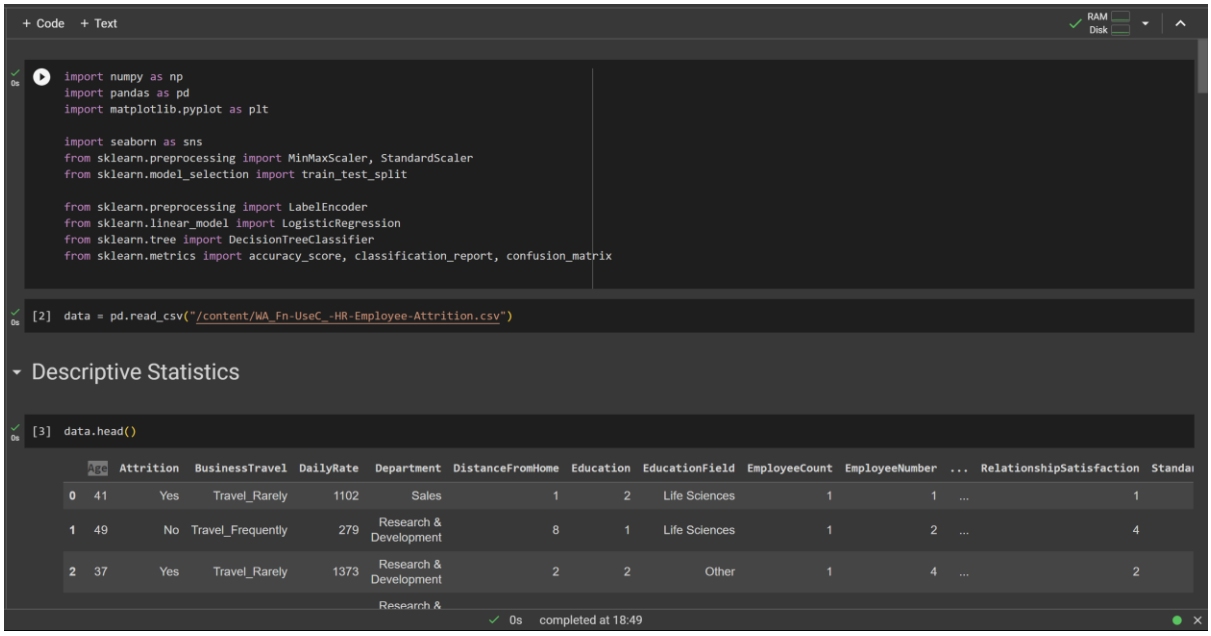
1.Download the Employee Attrition Dataset

https://www.kaggle.com/datasets/patelprashant/employee-attrition

2.Perfrom Data Preprocessing

3.Model Building using Logistic Regression and Decision Tree

4.Calculate Performance metrics



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

import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
[2] data = pd.read_csv("/content/WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

Descriptive Statistics

```
[3] data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	Standa
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	

0s completed at 18:49

Null Values

```
null = data.isnull()
nullval = null.sum()
print(nullval)
```

```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
```

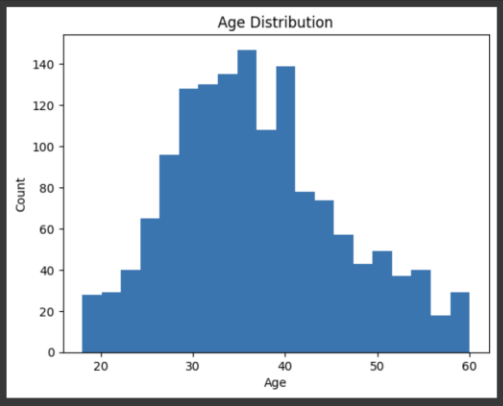
0s completed at 18:49

+ Code + Text

RAM Disk

Data visualisation

```
plt.hist(data['Age'], bins=20)
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Distribution')
plt.show()
```



0s completed at 18:49

+ Code + Text

RAM Disk

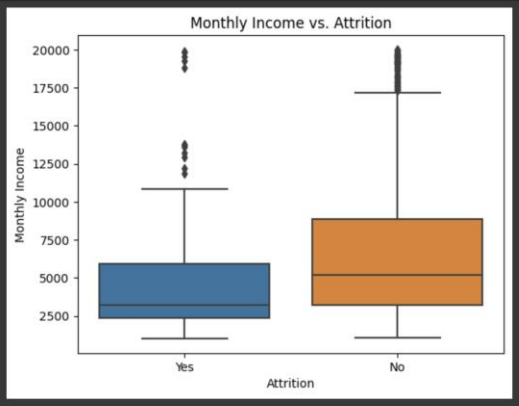
```
sns.countplot(x='Department', data=data)
plt.xlabel('Department')
plt.ylabel('Count')
plt.title('Department Distribution')
plt.xticks(rotation=45)
plt.show()
```

0s completed at 18:49

+ Code + Text

RAM Disk

```
sns.boxplot(x='Attrition', y='MonthlyIncome', data=data)
plt.xlabel('Attrition')
plt.ylabel('Monthly Income')
plt.title('Monthly Income vs. Attrition')
plt.show()
```



Splitting Dependent and Independent variables

0s completed at 18:49

+ Code + Text

RAM Disk

Splitting Dependent and Independent variables

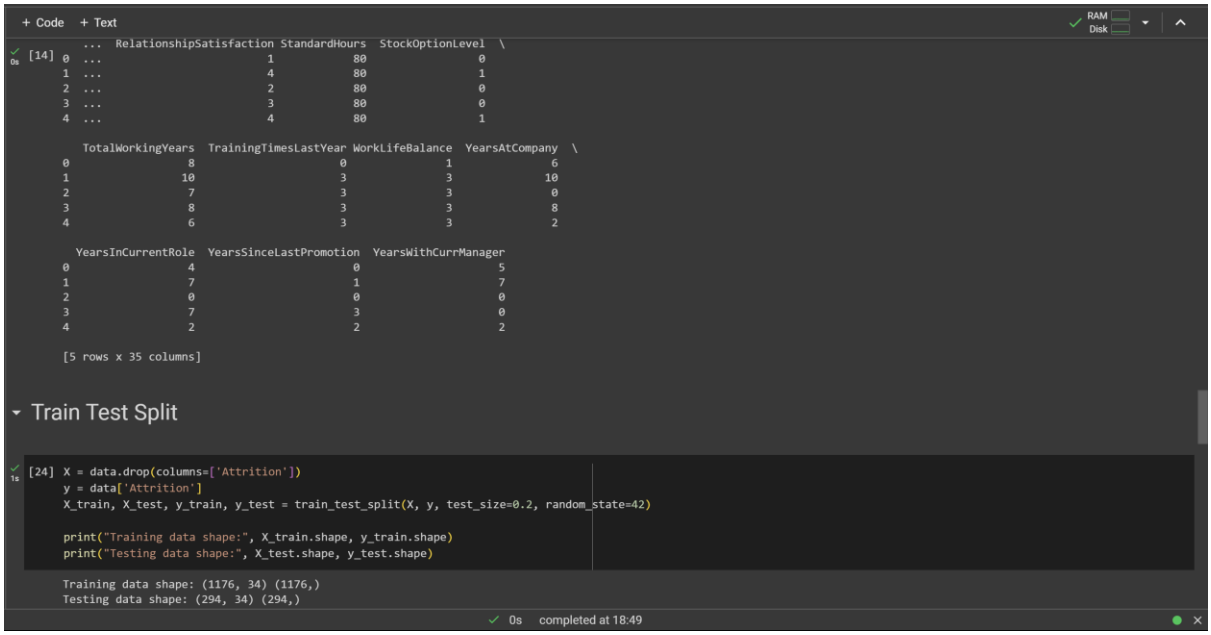
```
dep = data['Attrition']
indep = data.drop(columns=['Attrition'])
X = dep.values
y = indep.values
```

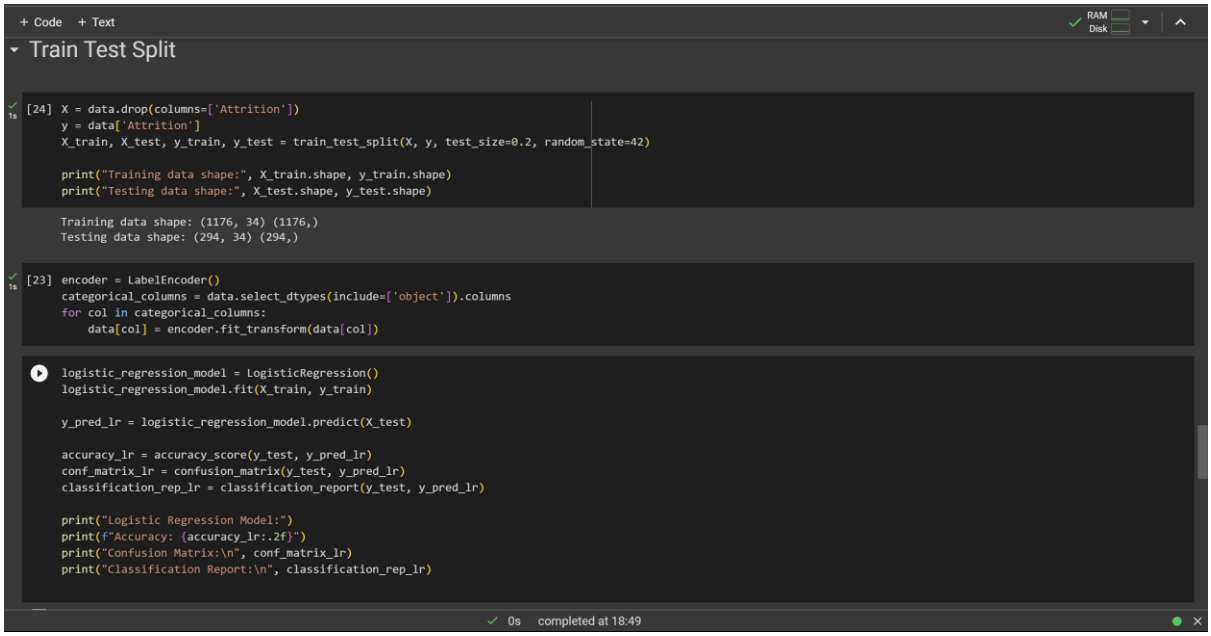
Feature Scaling

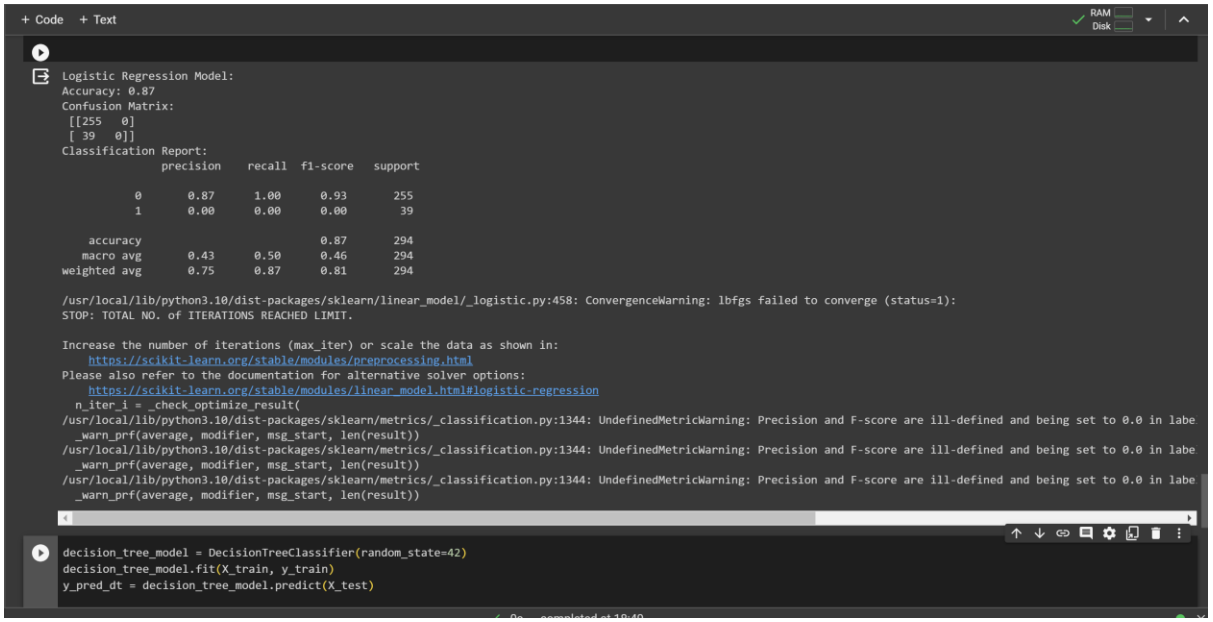
```
[14] columns_to_scale = ['Age', 'DailyRate']
min_max_scaler = MinMaxScaler()
data_min_max_scaled = data.copy()
data_min_max_scaled[columns_to_scale] = min_max_scaler.fit_transform(data[columns_to_scale])
print("Min-Max Scaled Data:")
print(data_min_max_scaled.head())
```

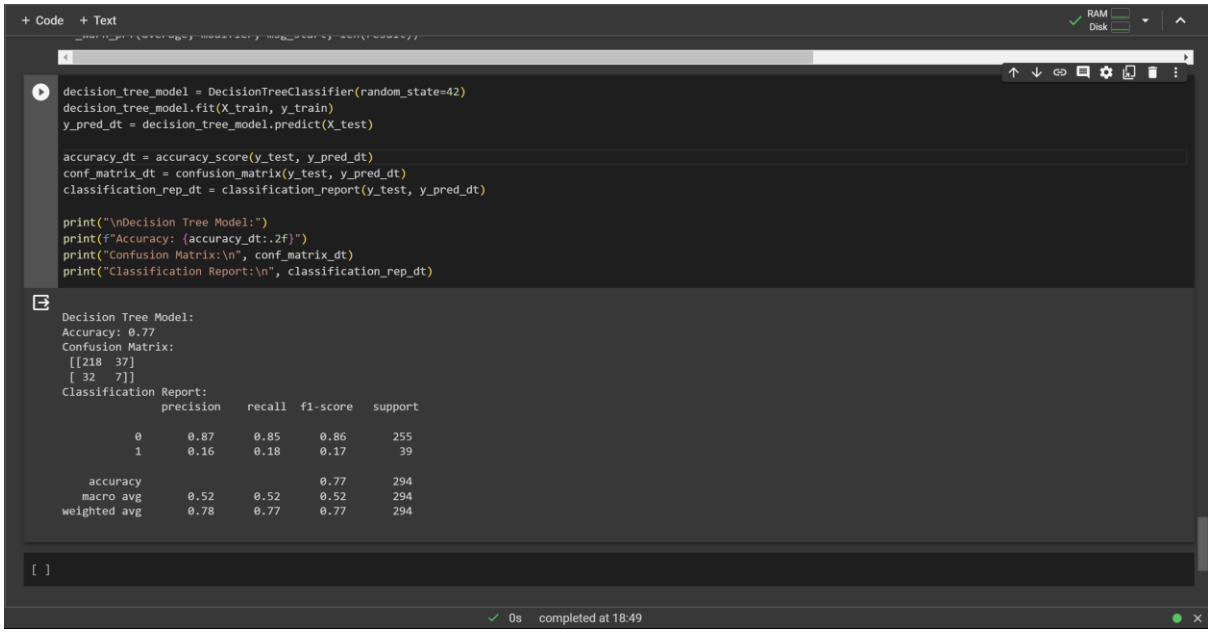
```
Min-Max Scaled Data:
        Age Attrition     BusinessTravel  DailyRate              Department  \
0  0.547619       Yes      Travel_Rarely   0.715820                   Sales
1  0.738095        No  Travel_Frequently   0.126700  Research & Development
2  0.452381       Yes      Travel_Rarely   0.909807  Research & Development
3  0.357143        No  Travel_Frequently   0.923407  Research & Development
4  0.214286        No      Travel_Rarely   0.350036  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumber  \
0                 1          2  Life Sciences              1               1
1                 8          1  Life Sciences              1               2
2                 2          2          Other              1               4
3                 3          4  Life Sciences              1               5
4                 2          1        Medical              1               7
```

0s completed at 18:49


+ Code + Text
RAM
Disk
[14]
... RelationshipSatisfaction StandardHours StockOptionLevel \
0 ... 1 80 0
1 ... 4 80 1
2 ... 2 80 0
3 ... 3 80 0
4 ... 4 80 1
TotalWorkingYears TrainingTimesLastYear WorkLifeBalance YearsAtCompany \
0 8 0 1 6
1 10 3 3 10
2 7 3 3 0
3 8 3 3 8
4 6 3 3 2
YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager
0 4 0 5
1 7 1 7
2 0 0 0
3 7 3 0
4 2 2 2
[5 rows x 35 columns]
Train Test Split
[24] X = data.drop(columns=['Attrition'])
y = data['Attrition']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Training data shape:", X_train.shape, y_train.shape)
print("Testing data shape:", X_test.shape, y_test.shape)
Training data shape: (1176, 34) (1176,)
Testing data shape: (294, 34) (294,)
0s completed at 18:49


+ Code + Text
RAM
Disk
Train Test Split
[24] X = data.drop(columns=['Attrition'])
y = data['Attrition']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Training data shape:", X_train.shape, y_train.shape)
print("Testing data shape:", X_test.shape, y_test.shape)
Training data shape: (1176, 34) (1176,)
Testing data shape: (294, 34) (294,)
[23] encoder = LabelEncoder()
categorical_columns = data.select_dtypes(include=['object']).columns
for col in categorical_columns:
data[col] = encoder.fit_transform(data[col])
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, y_train)
y_pred_lr = logistic_regression_model.predict(X_test)
accuracy_lr = accuracy_score(y_test, y_pred_lr)
conf_matrix_lr = confusion_matrix(y_test, y_pred_lr)
classification_rep_lr = classification_report(y_test, y_pred_lr)
print("Logistic Regression Model:")
print(f"Accuracy: {accuracy_lr:.2f}")
print("Confusion Matrix:\n", conf_matrix_lr)
print("Classification Report:\n", classification_rep_lr)
0s completed at 18:49


+ Code + Text
RAM
Disk
Logistic Regression Model:
Accuracy: 0.87
Confusion Matrix:
[[255 0]
[39 0]]
Classification Report:
precision recall f1-score support
0 0.87 1.00 0.93 255
1 0.00 0.00 0.00 39
accuracy 0.87 294
macro avg 0.43 0.50 0.46 294
weighted avg 0.75 0.87 0.81 294
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labe
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labe
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labe
_warn_prf(average, modifier, msg_start, len(result))
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)
y_pred_dt = decision_tree_model.predict(X_test)


+ Code + Text
RAM
Disk
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)
y_pred_dt = decision_tree_model.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
classification_rep_dt = classification_report(y_test, y_pred_dt)
print("\nDecision Tree Model:")
print(f"Accuracy: {accuracy_dt:.2f}")
print("Confusion Matrix:\n", conf_matrix_dt)
print("Classification Report:\n", classification_rep_dt)
Decision Tree Model:
Accuracy: 0.77
Confusion Matrix:
[[218 37]
[32 7]]
Classification Report:
precision recall f1-score support
0 0.87 0.85 0.86 255
1 0.16 0.18 0.17 39
accuracy 0.77 294
macro avg 0.52 0.52 0.52 294
weighted avg 0.78 0.77 0.77 294
[ ]
0s completed at 18:49