```
#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 pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
df = pd.read_csv('Employee-Attrition.csv')
```

```
# Check for missing values
print(df.isnull().sum())

# Check for outliers
import seaborn as sns
sns.boxplot(x = 'Age', y = 'Attrition', data = df)
sns.boxplot(x = 'DailyRate', y = 'Attrition', data = df)

# Check data types
print(df.dtypes)
```

```
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
Age                          int64
Attrition                   object
BusinessTravel              object
DailyRate                    int64
Department                  object
DistanceFromHome             int64
Education                    int64
EducationField              object
EmployeeCount                int64
EmployeeNumber               int64
EnvironmentSatisfaction      int64
Gender                      object
HourlyRate                   int64
JobInvolvement               int64
JobLevel                     int64
JobRole                     object
JobSatisfaction              int64
MaritalStatus               object
MonthlyIncome                int64
MonthlyRate                  int64
NumCompaniesWorked           int64
Over18                      object
OverTime                    object
PercentSalaryHike            int64
PerformanceRating            int64
RelationshipSatisfaction     int64
StandardHours                int64
StockOptionLevel             int64
TotalWorkingYears            int64
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Handle missing values
df = df.dropna()

# Handle outliers
df = df[(df['Age'] < 70) & (df['DailyRate'] < 200000)]

# Encode categorical features
df['Gender'] = df['Gender'].map({'Male': 0, 'Female': 1})
df['Department'] = df['Department'].map({'Sales': 0, 'Engineering': 1, 'Marketing': 2})

# Scale the numeric features
scaler = StandardScaler()
X = df[['Age', 'DailyRate', 'Gender', 'Department']]
X = scaler.fit_transform(X)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, df['Attrition'], test_size=0.25, random_state=42)

# Convert the NumPy arrays to Pandas DataFrames
X_train_df = pd.DataFrame(X_train)
X_test_df = pd.DataFrame(X_test)

# Save the preprocessed data to CSV files
X_train_df.to_csv('X_train.csv', index=False)
X_test_df.to_csv('X_test.csv', index=False)
y_train.to_csv('y_train.csv', index=False)
y_test.to_csv('y_test.csv', index=False)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1047: RuntimeWarning: invalid value encountered in divide
  updated_mean = (last_sum + new_sum) / updated_sample_count
/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1052: RuntimeWarning: invalid value encountered in divide
  T = new_sum / new_sample_count
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1072: RuntimeWarning: invalid value encountered in divide
  new_unnormalized_variance -= correction**2 / new_sample_count
```

```
X_train_df.to_csv
```

```
<bound method NDFrame.to_csv of             0         1   2   3
0   -0.170973 -1.235942 NaN NaN
1   -0.503476 -0.487829 NaN NaN
2    2.489050  1.055620 NaN NaN
3   -0.946813  1.676977 NaN NaN
4    1.048204  0.807078 NaN NaN
..        ...       ...  ..  ..
329 -0.946813 -1.233456 NaN NaN
330 -0.170973  1.023310 NaN NaN
331 -0.835979 -0.823361 NaN NaN
332 -0.281807 -0.239286 NaN NaN
333 -0.392642 -1.119127 NaN NaN

[334 rows x 4 columns]>
```

```
X_test_df.to_csv
```

```
<bound method NDFrame.to_csv of             0         1   2   3
0    0.494033 -1.352757 NaN NaN
1    1.269873 -0.229345 NaN NaN
2    0.494033 -0.385926 NaN NaN
3    1.824044  0.913951 NaN NaN
4   -0.835979  1.018339 NaN NaN
..        ...       ...  ..  ..
107 -0.170973 -0.092646 NaN NaN
108  0.272364  0.235430 NaN NaN
109 -0.725144  1.338959 NaN NaN
110  0.937370 -1.228485 NaN NaN
111 -0.946813  0.923893 NaN NaN

[112 rows x 4 columns]>
```

```
y_train.to_csv
```

```
<bound method NDFrame.to_csv of 1281    Yes
935      No
70       No
1369    Yes
433      No
       ...
374      No
888      No
1172     No
1446     No
363     Yes
Name: Attrition, Length: 334, dtype: object>
```

```
y_test.to_csv
```

```
<bound method NDFrame.to_csv of 951      No
1204    Yes
403      No
1396    Yes
265      No
       ...
1167    Yes
33      Yes
167      No
504     Yes
1337     No
Name: Attrition, Length: 112, dtype: object>
```

```
#3
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression


# Load the preprocessed training data
X_train = pd.read_csv('X_train.csv')
y_train = pd.read_csv('y_train.csv')['Attrition']

# Load the preprocessed test data
```

```
X_test = pd.read_csv('X_test.csv')
y_test = pd.read_csv('y_test.csv')['Attrition']


# Create a logistic regression model
logistic_model = LogisticRegression(random_state=42)


import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.impute import SimpleImputer

# Impute the missing values in the Age feature
imputer = SimpleImputer(strategy='mean')
df['Age'] = imputer.fit_transform(df[['Age']])

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(df[['Age', 'DailyRate', 'Gender', 'Department']], df['Attrition'], test_size=0.25, random

# Create a logistic regression model
logistic_model = LogisticRegression(random_state=42)

# Train the logistic regression model
logistic_model.fit(X_train, y_train)

# Evaluate the logistic regression model on the test data
y_pred = logistic_model.predict(X_test)
accuracy = logistic_model.score(X_test, y_test)
print('Accuracy:', accuracy)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-23-63d8f7aba945> in <cell line: 17>()
     15 
     16 # Train the logistic regression model
---> 17 logistic_model.fit(X_train, y_train)
     18 
     19 # Evaluate the logistic regression model on the test data

                              4 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in _assert_all_finite(X,
allow_nan, msg_dtype, estimator_name, input_name)
    159                 "#estimators-that-handle-nan-values"
    160             )
--> 161         raise ValueError(msg_err)
    162 
    163 

ValueError: Input X contains NaN.
LogisticRegression does not accept missing values encoded as NaN natively. For supervised learning,
you might want to consider sklearn.ensemble.HistGradientBoostingClassifier and Regressor which accept
missing values encoded as NaNs natively. Alternatively, it is possible to preprocess the data, for
instance by using an imputer transformer in a pipeline or drop samples with missing values. See
https://scikit-learn.org/stable/modules/impute.html You can find a list of all estimators that handle
NaN values at the following page: https://scikit-learn.org/stable/modules/impute.html#estimators-that-
handle-nan-values
```

```
from sklearn.impute import SimpleImputer

# Impute the missing values in the Age feature of the train set
imputer = SimpleImputer(strategy='mean')
X_train['Age'] = imputer.fit_transform(X_train[['Age']])

# Impute the missing values in the Age feature of the test set
imputer = SimpleImputer(strategy='mean')
X_test['Age'] = imputer.fit_transform(X_test[['Age']])
```