1) Perfrom Data Preprocessing

```
#Importing necessary Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv("/content/WA_Fn-UseC_-HR-Employee-Attrition.csv")

# Drop unused columns
data = data.drop(["EmployeeCount", "EmployeeNumber", "StandardHours", "Over18"], axis=1)

# Define the numerical and categorical feature columns
numerical_features = ["Age", "DailyRate", "HourlyRate", "MonthlyIncome", "PercentSalaryHike"]
categorical_features = ["BusinessTravel", "Department", "EducationField", "Gender", "JobRole", "MaritalStatus", "OverTime"]

# Split into features (X) and target (y)
X = data.drop("Attrition", axis=1)
y = data["Attrition"]

# Perform one-hot encoding for categorical features
X_encoded = pd.get_dummies(X, columns=categorical_features)

# Scale numerical features
scaler = StandardScaler()
X_encoded[numerical_features] = scaler.fit_transform(X_encoded[numerical_features])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=0)
```

2) Model Building

i) LOGISTIC REGRESSION

```
logistic_model = LogisticRegression(max_iter=1000, random_state=0)

logistic_model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000, random_state=0)
```

ii) DecisionTreeClassifier

```
decision_tree_model = DecisionTreeClassifier( random_state=0)

decision_tree_model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

iii) RandomForestClassifier

```
random_forest_model = RandomForestClassifier( random_state=0)

random_forest_model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=0)
```

3) Hyper Parameter Tuning

```
param_grid = {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10]}

logistic_grid = GridSearchCV(LogisticRegression(random_state=0, max_iter=2000), param_grid, cv=5, scoring='accuracy')

logistic_grid.fit(X_train, y_train)

best_params = logistic_grid.best_params_
best_logistic_model = logistic_grid.best_estimator_
```

4) Calculating Performance Metrics

i) Logistic Regression

```
y_pred_logistic = best_logistic_model.predict(X_test)
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)

print("Logistic Regression Accuracy:", accuracy_logistic)
print("Logistic Regression Classification Report:\n", classification_report(y_test, y_pred_logistic))
print("Logistic Regression Confusion Matrix:\n", confusion_matrix(y_test, y_pred_logistic))
```

```
Logistic Regression Accuracy: 0.8979591836734694
Logistic Regression Classification Report:
              precision    recall  f1-score   support

          No       0.90      0.98      0.94       245
         Yes       0.85      0.47      0.61        49

    accuracy                           0.90       294
   macro avg       0.88      0.73      0.77       294
weighted avg       0.89      0.90      0.89       294

Logistic Regression Confusion Matrix:
 [[241   4]
 [ 26  23]]
```

ii) Decision Tree

```
y_pred_tree = best_tree_model.predict(X_test)
accuracy_tree = accuracy_score(y_test, y_pred_tree)

print("\nDecision Tree Accuracy:", accuracy_tree)
print("Decision Tree Classification Report:\n", classification_report(y_test, y_pred_tree))
print("Decision Tree Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tree))
```

```
Decision Tree Accuracy: 0.8639455782312925
Decision Tree Classification Report:
              precision    recall  f1-score   support

          No       0.89      0.96      0.92       245
         Yes       0.65      0.41      0.50        49

    accuracy                           0.86       294
   macro avg       0.77      0.68      0.71       294
weighted avg       0.85      0.86      0.85       294

Decision Tree Confusion Matrix:
 [[234  11]
 [ 29  20]]
```

iii) Random Forest

```
y_pred_random_forest = random_forest_model.predict(X_test)
accuracy_random_forest = accuracy_score(y_test, y_pred_random_forest)

print("Random Forest Accuracy:", accuracy_random_forest)
print("\nRandom Forest Classification Report:\n", classification_report(y_test, y_pred_random_forest))
print("Random Forest Confusion Matrix:\n", confusion_matrix(y_test, y_pred_random_forest))
```

```
Random Forest Accuracy: 0.8571428571428571

Random Forest Classification Report:
              precision    recall  f1-score   support

          No       0.86      0.99      0.92       245
         Yes       0.82      0.18      0.30        49

    accuracy                           0.86       294
   macro avg       0.84      0.59      0.61       294
weighted avg       0.85      0.86      0.82       294

Random Forest Confusion Matrix:
 [[243   2]
 [ 40   9]]
```