# Machine Learning Tutorial Report

**Assignment:** Individual Assignment – Machine Learning Tutorial

**Module:** Machine Learning and Neural Networks

**Name:** Deepak Raj Manickam

**Student ID:** 23070139; **Student Email ID:** dm24aav@herts.ac.uk

**GitHub Repository:** **https://github.com/deepakraj-04/Machine-Learning-and-Neural-Networks.git**

**Introduction:**

Machine learning has transformed the field of data analysis and predictive modelling. Machine learning enables the computers to learn from the data and make predictions. It uses algorithms to identify patterns and improve the performance of the model. Machine learning is offering powerful tools for pattern recognition and decision making. Machine is used in areas like finance, healthcare and automation.

Random Forest is a widely used learning technique to enhance predictive accuracy and prevent overfitting. Random Forest is particularly useful in classification and regression tasks and operates by constructing multiple decision trees during training.

This tutorial provides a step by step guide on implementing the random forest classifier using scikit-learn library in python. This tutorial covers by explaining random forest algorithm, data preprocessing, model training, evaluation, feature importance analysis and visualization of results using plots.

**What is Random Forest?**

Random Forest is a powerful machine learning algorithm which consists of multiple decision trees. It operates by constructing multiple decision trees. For classification it makes prediction by using majority voting. For regression it makes prediction by averaging outputs. A single decision tree can overfit which means it might not perform well on the new data. Random forest avoids this issue by creating multiple decision trees. It trains each decision tree with a random subset of the data and choose random features to split on. This randomness can help the model to learn different patterns of the data by making it more accurate and less likely to overfit.

**Why we use Random Forest?**

Random Forest is good at making accurate predictions, handles noisy data and it can work with large datasets with many features. It also reduces overfitting by combining predictions from multiple decision tree.

**Some key reasons to use random forest:**

- It provides better accuracy than a single decision tree by combining multiple decision trees.
- The model can handle missing values.
- It can perform well even when there are more features which makes it more suitable for more complex datasets.
- It can be applied for both classification and regression problems.
- Random Forest can handle noisy data.
- A single decision tree can overfit but random forest train multiple decision trees by making it more accurate model and less like to overfit.

**Dataset Description:**

For this tutorial we are using IBM HR Employee Attrition Dataset. This dataset contains employee details and their attrition status. This dataset contains of 1470 rows and 35 columns which includes variables like:

1. Age
2. Attrition
3. BusinessTravel
4. DailyRate
5. Department
6. DistanceFromHome
7. Education
8. EducationField
9. EmployeeCount
10. EmployeeNumber
11. EnvironmentSatisfaction
12. Gender
13. HourlyRate
14. JobInvolvement
15. JobLevel

16. JobRole
17. JobSatisfaction
18. MaritalStatus
19. MonthlyIncome
20. MonthlyRate
21. NumCompaniesWorked
22. Over18
23. OverTime
24. PercentSalaryHike
25. PerformanceRating
26. RelationshipSatisfaction
27. StandardHours
28. StockOptionLevel
29. TotalWorkingYears
30. TrainingTimesLastYear
31. WorkLifeBalance
32. YearsAtCompany
33. YearsInCurrentRole
34. YearsSinceLastPromotion
35. YearsWIthCurrManager

**Dataset Preprocessing:**

1. **Data Cleaning:** Remove duplicates and missing values.
2. **Encoding Categorical Variables:** Convert categorical variables into numerical form.

**Implementing Random Forest in Python:**

In this tutorial we are using scikit-learn for model training and evaluation.

1. **Importing necessary libraries**

   **Code:**

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc
```

2. **Loading and displaying the dataset**

   **Code:**

```
# Load the dataset
IBM_df = pd.read_csv("/Users/deepakraj/Desktop/WA_Fn-UseC_-HR-Employee-Attrition.csv")

# Display first few rows of the dataset
IBM_df.head()
```

3. **Checking basic dataset information**

   **Code:**

```
# Display basic information of the dataset
IBM_df.info()
```

4. **Handling missing values**

   **Code:**

```
# Drop rows with missing values
IBM_df.dropna(inplace=True)
```

5. **Encoding categorical variables and defining target and features**

   **Code:**

```
# Convert categorical variables to numeric
IBM_df = pd.get_dummies(IBM_df, drop_first=True)
```

```
# Define target and features
y = IBM_df['Attrition_Yes']
x = IBM_df.drop('Attrition_Yes', axis=1)
```

6. **Splitting data into training and testing sets and training the random forest model**

   **Code:**

```
# Split into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42, stratify=y)
```

```
# Train Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
rf_model.fit(x_train, y_train)
```

7. **Making predictions and evaluating the model**

**Code:**

```
# Make predictions
y_pred = rf_model.predict(x_test)
```

```
# Evaluate model
accuracy = accuracy_score(y_test, y_pred)
confusionmatrix = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f"Random Forest Accuracy: {accuracy:.4f}")
print("Confusion Matrix:\n", confusionmatrix)
print("Classification Report:\n", report)
```

**Output:**

```
Random Forest Accuracy: 0.8401
Confusion Matrix:
 [[244   3]
 [ 44   3]]
Classification Report:
               precision    recall  f1-score   support

       False       0.85      0.99      0.91       247
        True       0.50      0.06      0.11        47

    accuracy                           0.84       294
   macro avg       0.67      0.53      0.51       294
weighted avg       0.79      0.84      0.78       294
```

**Plot 1: Feature Importance**

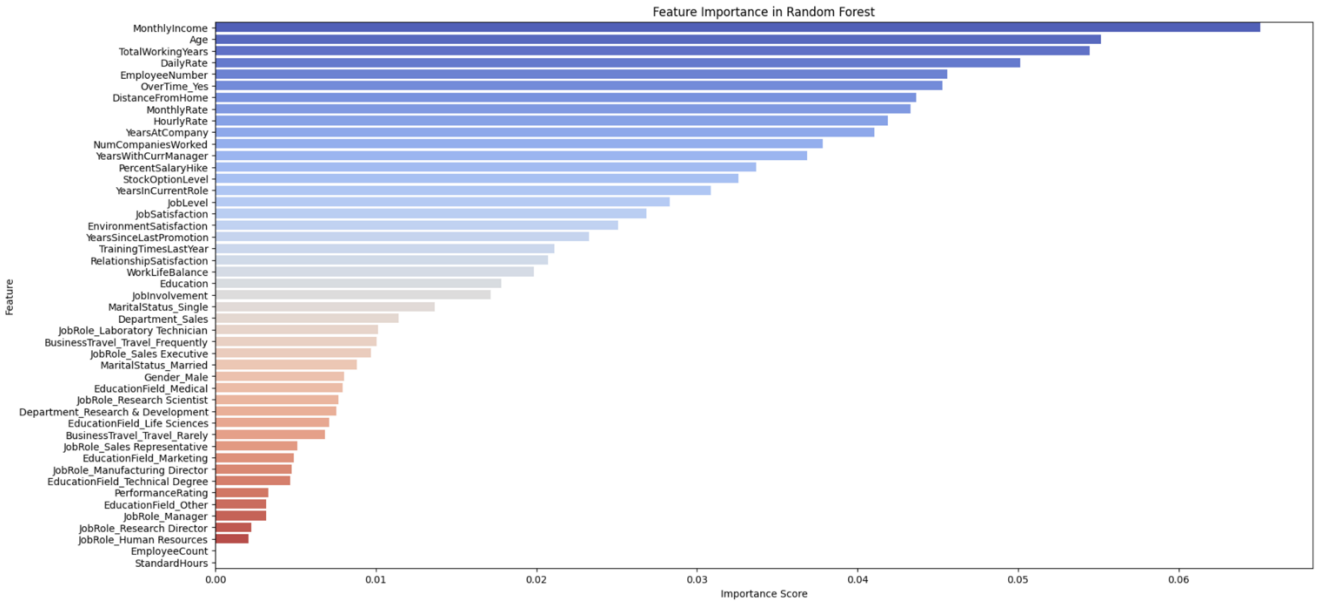
This plot shows us the importance of different features in the random forest model. The x-axis represents importance score and the y-axis represents feature from least to most important. Blue bars indicate highly important features. Red bars indicate least important features. This helps in choosing the most useful features for better prediction. The model will focus on more important features to improve accuracy. This data analysis easier to understand and use.

**Code:**

```
feature_importance = pd.Series(rf_model.feature_importances_, index=x.columns).sort_values(ascending=False)

plt.figure(figsize=(20,10))
sns.barplot(x=feature_importance.values, y=feature_importance.index, hue=feature_importance.index, palette="coolwarm", legend=False)
"""
It creates a bar plot of feature importance from Random Forest Model.
"""
plt.title("Feature Importance in Random Forest")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.show()
```

**Output:**




**Plot 2: ROC Curve**

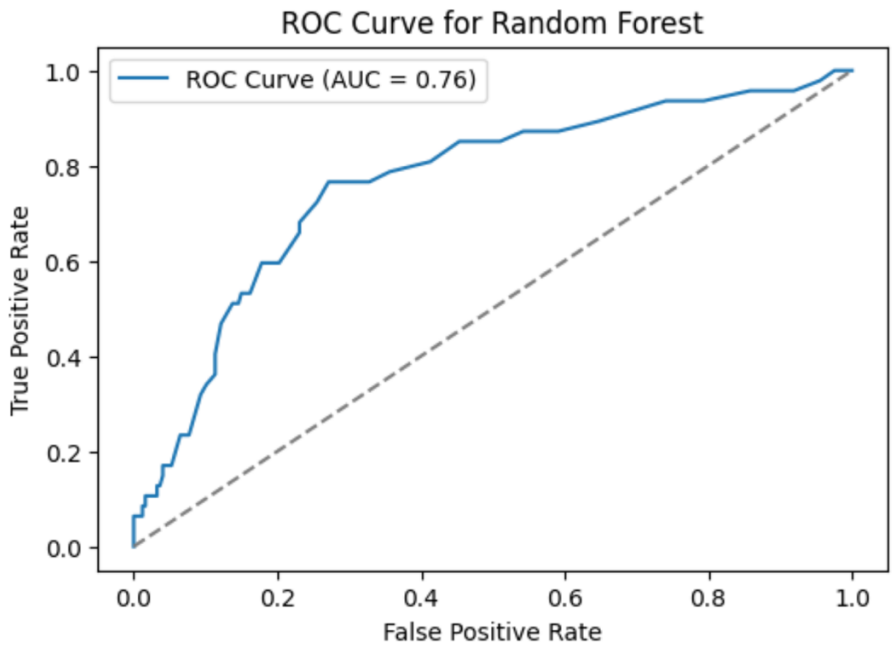
This plot is a ROC Curve for random forest model. The x-axis represents false positive rate. The y-axis represents true positive rate. The blue line represents the model performance. The dashed line represents random guessing. The AUC is 0.76 which means that the model is good at making predictions. A higher AUC makes a better model. This graph shows how well the model separates the different classes.

**Code:**

```
fpr, tpr, _ = roc_curve(y_test, rf_model.predict_proba(x_test)[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')
"""
It plot the ROC curve for Random Forest Model and it displays the AUC score.
"""
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Random Forest')
plt.legend()
plt.show()
```

**Output:**



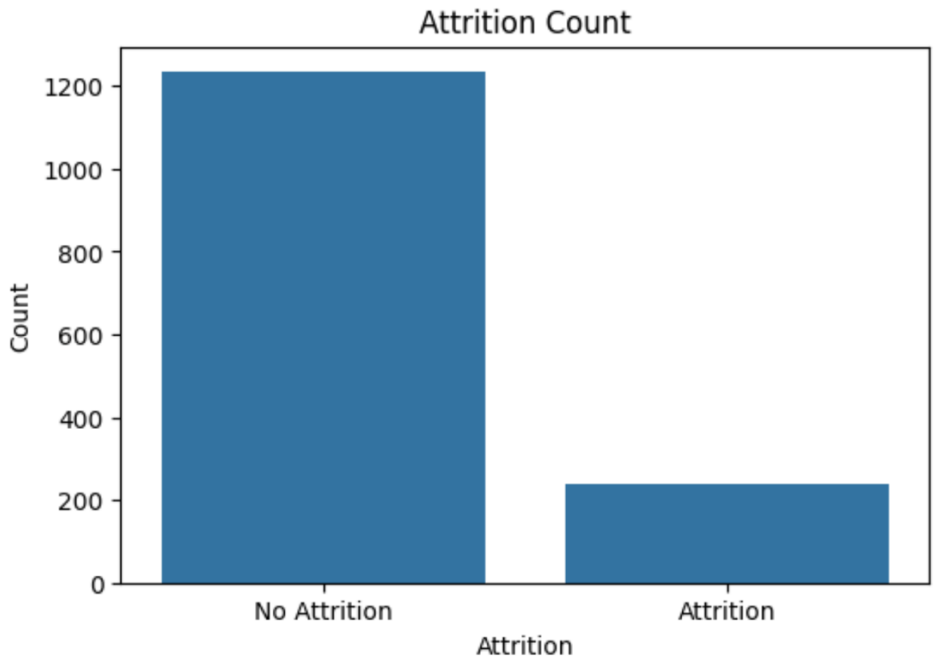

**Plot 3: Attrition Count**

This graph is a count plot which shows us the attrition categories. The x-axis represents two categories No Attrition and Attrition. The y-axis represents the number of employees in each category. The No Attrition category has more count that the Attrition category. This graph shows most of the employees stayed in the company while fewer employees left the company. This graph helps us to understand how many employees stayed or left.

**Code:**

```
plt.figure(figsize=(6,4))
sns.countplot(x=y)
"""
It creates a count plot to visualize the distribution of attrition categories.
"""
plt.title("Attrition Count")
plt.xticks([0, 1], ["No Attrition", "Attrition"])
plt.xlabel("Attrition")
plt.ylabel("Count")
plt.show()
```

**Output:**



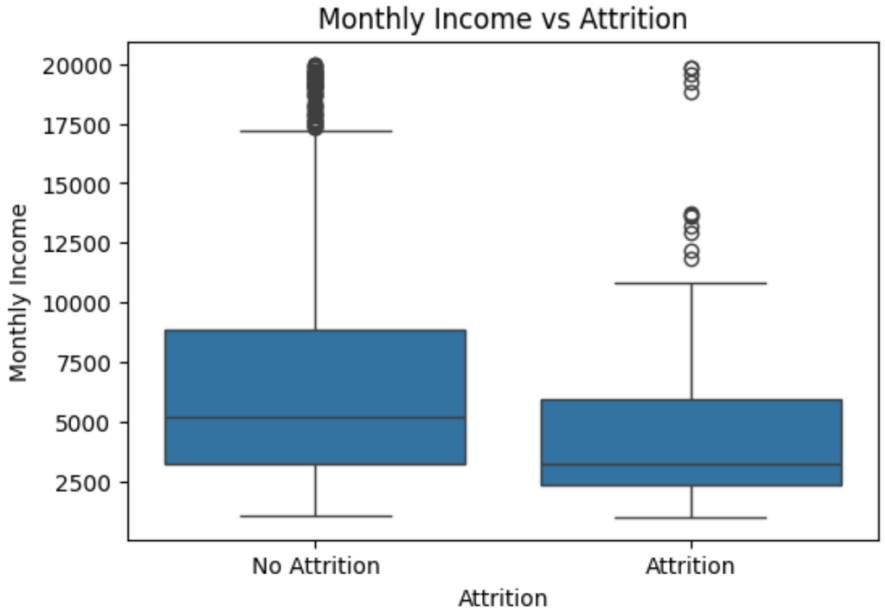

**Plot 4: Box Plot**

This box plot represents the relationship between monthly income and attrition. Employees (Attrition) who has left the company had lower salaries. Employees who stayed in the company has a wide range of incomes with some earnings. There are some dots above the box represents higher salary. This graph shows us that the employees pay might affect whether they stay or leave.

**Code:**

```
plt.figure(figsize=(6, 4))
sns.boxplot(x=y, y=IBM_df["MonthlyIncome"])
"""
It creates a box plot to visualize the relationship between monthly income vs attrition.
"""
plt.title("Monthly Income vs Attrition")
plt.xticks([0, 1], ["No Attrition", "Attrition"])
plt.xlabel("Attrition")
plt.ylabel("Monthly Income")
plt.show()
```

**Output:**




**Conclusion:**

Random Forest is a powerful machine learning algorithm suitable for both classification and regression tasks. This tutorial has demonstrated its implementation using IBM HR Attrition dataset which covers data processing, model training, evaluation and visualizations.

**References:**


1. IBM HR Attrition Dataset (Kaggle)
2. Official Scikit-Learn Documentation – Random Forest
3. Alpaydin, E., 2021. *Machine learning*. MIT press.
4. Kramer, O. and Kramer, O., 2016. Scikit-learn. *Machine learning for evolution strategies*, pp.45-53.
5. Jaiswal, J.K. and Samikannu, R., 2017, February. Application of random forest algorithm on feature subset selection and classification and regression. In *2017 world congress on computing and communication technologies (WCCCT)* (pp. 65-68). Ieee.
6. Salman, H.A., Kalakech, A. and Steiti, A., 2024. Random forest algorithm overview. *Babylonian Journal of Machine Learning*, *2024*, pp.69-79.