Configuration Manual

MSc Research Project
Data Analytics

Nistala Maneesh
Student ID: x23132914

School of Computing
National College of Ireland

Supervisor: Mr. Hicham Rifai

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nistala Maneesh……………………………………………………………
Student ID:	……x23132914……………………………………………………………
Programme:	…..MSc Data Analytics…………….. **Year:** ………2024…………………..
Module:	Research Project……………………………………………………………
Lecturer:	Mr. Hicham Rifai………………………………………………………….
Submission Due Date:	………12/12/2024………………………………………………………………
Project Title:	Predicting Employee Attrition Using Machine Learning in Tech Industry: A Methodological Approach………………………
Word Count:	……………1112………………………… **Page Count: 12**…………………..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ………Nistala Maneesh…………………………………………………….

Date: ………12/12/2024…………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Nistala Maneesh
X23132914

Configuration Manual
Section 1: Project Overview

This project aims at developing a machine learning model to identify the likelihood of employees leaving the organization. The dataset comprises of variables like age, income, work-life balance and other characteristics of the organization. Features such as Logistic Regression, Random Forest, Decision Tree, SVM and KNN are used to assess their predictive power.

Purpose of the Configuration Manual

This manual is designed to:

1. Guide users through the setup and execution of machine learning models for predicting employee attrition.
2. Provide step-by-step instructions for dataset preparation, model training, and evaluation.
3. Help users interpret model results and optimize their configurations.
4. Serve as a reference for troubleshooting common issues.

The manual covers:

- Environment setup and system requirements.
- Dataset preparation and preprocessing.
- Training and evaluating multiple machine learning models.
- Interpretation of results and visualizations.
- Performance comparison and model selection.

System Requirements

Hardware Requirements

1. **Minimum Specifications:**

- **CPU:** Multi-core processor.
- **RAM:** 8GB system memory.
- **Storage:** 10GB free disk space.

2. **Recommended Specifications:**

- **CPU:** Quad-core processor.
- **RAM:** 16GB or higher for faster processing.
- **Storage:** 50GB free disk space.

Software Requirements

1. **Operating System:**
- Compatible with Windows, macOS, or Linux.
2. **Development Environment:**
- Python 3.8 or higher installed.
- Jupyter Notebook or any Python-supported IDE.
3. **Required Libraries:**
- **Core Libraries:** scikit-learn 1.2.2, pandas 1.5.3, numpy 1.24.3.
- **Visualization:** matplotlib 3.7.1, seaborn 0.12.2.
- **Additional Tools:** scipy 1.10.1.

Environment Setup Requirements

Environment Preparation

1. Install Python 3.8+ from [Python.org](https://www.python.org).
2. Install required libraries using pip:

```
pip install pandas numpy scikit-learn matplotlib seaborn scipy
```

Dataset Preparation

1. Organize the dataset in a dedicated project folder:
   - **IBM_HR_Analytics.csv:** Contains the employee data.
2. Ensure the dataset has no corrupted or missing files.

Project Structure Setup

1. **Main Project Directory:**
   Create the following folder structure:

```
Employee_Attrition_Prediction
  ├── Dataset
  │   └── IBM_HR_Analytics.csv
  ├── Notebooks
  │   └── analysis_notebook.ipynb
  ├── Results
  │   ├── Confusion_Matrices
  │   ├── ROC_Curves
  │   └── Summary
```

2. **Results Organization:**

- **Confusion_Matrices:** Store confusion matrix visualizations for each model.
- **ROC_Curves:** Save ROC curve plots for all models.
- **Summary:** Include the performance summary table as a CSV file.

System Verification

Essential Checks

1. **Python Environment:**
   - Confirm Python 3.8+ is installed.
   - Validate the installation of required libraries.
2. **Dataset Accessibility:**
   - Ensure the dataset file path is correct.
   - Verify read/write permissions in the project directory.
3. **RAM and CPU Usage:**
   - Monitor memory usage during model training.
   - Close unnecessary applications to free up system resources.

Troubleshooting Guide

Common Issues and Solutions:

1. **Library Installation Errors:**
   - Ensure pip is updated:

     pip install --upgrade pip
   - Reinstall missing libraries using pip.
2. **Dataset Loading Issues:**
   - Verify the dataset path is correct.
   - Ensure the dataset is in .csv format and uncorrupted.
3. **Model Training Errors:**
   - Reduce dataset size or batch size to resolve memory issues.
   - Ensure all features are properly encoded before training.
4. **Visualization Errors:**
   - Confirm matplotlib and seaborn are installed and up to date.
   - Verify that the dataset has no missing or NaN values.

Section 2: Code Setup and Execution

1. Import Libraries

The necessary libraries for data manipulation, machine learning, and visualization are imported using these commands:

```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.inspection import permutation_importance
from scipy.stats import ttest_ind
```

2. Load the Dataset

The IBM HR Analytics dataset is loaded, and the first few rows are displayed to confirm successful loading.

```
# Load the dataset
df = pd.read_csv(r"C:\Users\nista\OneDrive\Desktop\Rsearch Project\IBM_HR_Analytics.csv")

# Display dataset information
print("Dataset Head:")
print(df.head())
print("\nDataset Description:")
print(df.describe())
```

The following is a presentation of the code results after its execution, including the dataset head and other information that can be extracted from the dataset.

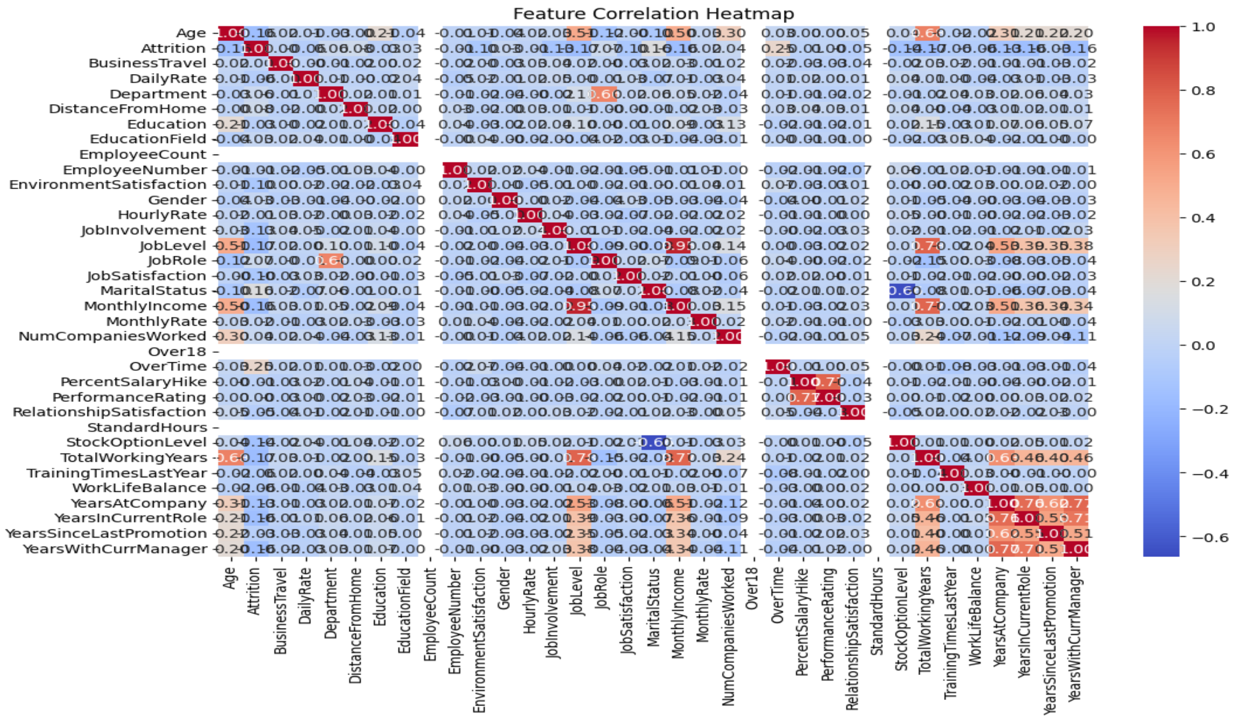
```
Dataset Head:
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development
```

3. Data Visualization: Correlation Heatmap

This code is used to generate the heatmap, which is a major tool in this modelling

```
# Encode categorical variables for correlation heatmap
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    df[col] = LabelEncoder().fit_transform(df[col])

# Correlation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```

Add the generated heatmap to highlight feature relationships.



4. Data Preprocessing

Categorical variables are encoded, missing values are handled, and features are split from the target variable.

```python
# Data preprocessing
# Encode categorical variables for model use
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    df[col] = LabelEncoder().fit_transform(df[col])

# Encoding target variable
df['Attrition'] = LabelEncoder().fit_transform(df['Attrition'])

# Handling missing values if any
df.fillna(df.median(), inplace=True)

# Feature-target split
X = df.drop('Attrition', axis=1)
y = df['Attrition']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

5. Standardization for Specific Models

Features are standardized for models like Logistic Regression, SVM, and KNN.

```python
# Standardize features for Logistic Regression, SVM, and KNN
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

6. Train Models and Perform Cross-Validation

Each model is evaluated using 5-fold cross-validation with ROC AUC scores.

```python
# Models
models = {
    "Logistic Regression": LogisticRegression(solver='saga', max_iter=1000, random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "KNN": KNeighborsClassifier()
}

# Cross-validation to collect multiple ROC AUC scores
cv_results = {}
for model_name, model in models.items():
    if model_name in ["SVM", "KNN", "Logistic Regression"]:  # Use scaled data for these models
        roc_aucs = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring='roc_auc')
    else:  # Use raw data for others
        roc_aucs = cross_val_score(model, X_train, y_train, cv=5, scoring='roc_auc')
    cv_results[model_name] = roc_aucs
    print(f"{model_name} ROC AUCs from Cross-Validation: {roc_aucs}")
    print(f"Mean ROC AUC: {np.mean(roc_aucs):.4f}, Std Dev: {np.std(roc_aucs):.4f}")
```

The image below show cross-validation ROC AUC results for each model together with their accuracy.

Summary of Results:

	Model	Accuracy	ROC AUC
0	Logistic Regression	0.874150	0.804720
1	Random Forest	0.829932	0.803816
2	Decision Tree	0.765306	0.593290
3	SVM	0.863946	0.810923
4	KNN	0.846939	0.665949

7. Model Training and Evaluation

Models are trained on the training set and evaluated on the test set for accuracy, ROC AUC, classification report, and confusion matrix.

```python
# Train and evaluate models on the test set
results = {}
feature_importance = {}

for model_name, model in models.items():
    if model_name in ["SVM", "KNN", "Logistic Regression"]:  # Use scaled data for these models
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
        y_proba = model.predict_proba(X_test_scaled)[:, 1]

        # Feature importance
        if model_name == "Logistic Regression":
            importance = np.abs(model.coef_[0])  # Coefficients represent importance
            feature_importance[model_name] = importance
        elif model_name == "KNN":
            perm_importance = permutation_importance(model, X_test_scaled, y_test)
            importance = perm_importance.importances_mean
            feature_importance[model_name] = importance
    else:  # Use raw data for others
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        y_proba = model.predict_proba(X_test)[:, 1]

        # Feature importance
        importance = model.feature_importances_  # Native feature importance
        feature_importance[model_name] = importance

    # Metrics
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_proba)
    report = classification_report(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
```

Each of the model confusion matrices and classification reports are well shown by this code.

8. Summary of Results

For the purpose of comparing the models accuracy, the following code was used. The summary of the summary results is shown below.

```python
# Summary Table
summary_table = pd.DataFrame({
    "Model": list(results.keys()),
    "Accuracy": [results[model]["Accuracy"] for model in results],
    "ROC AUC": [results[model]["ROC AUC"] for model in results]
})
print("Summary of Results:")
print(summary_table)
```

The following is the summary table for all of the model. This is used so as to make easy to retrieve the values obtained in each of the models at ease.

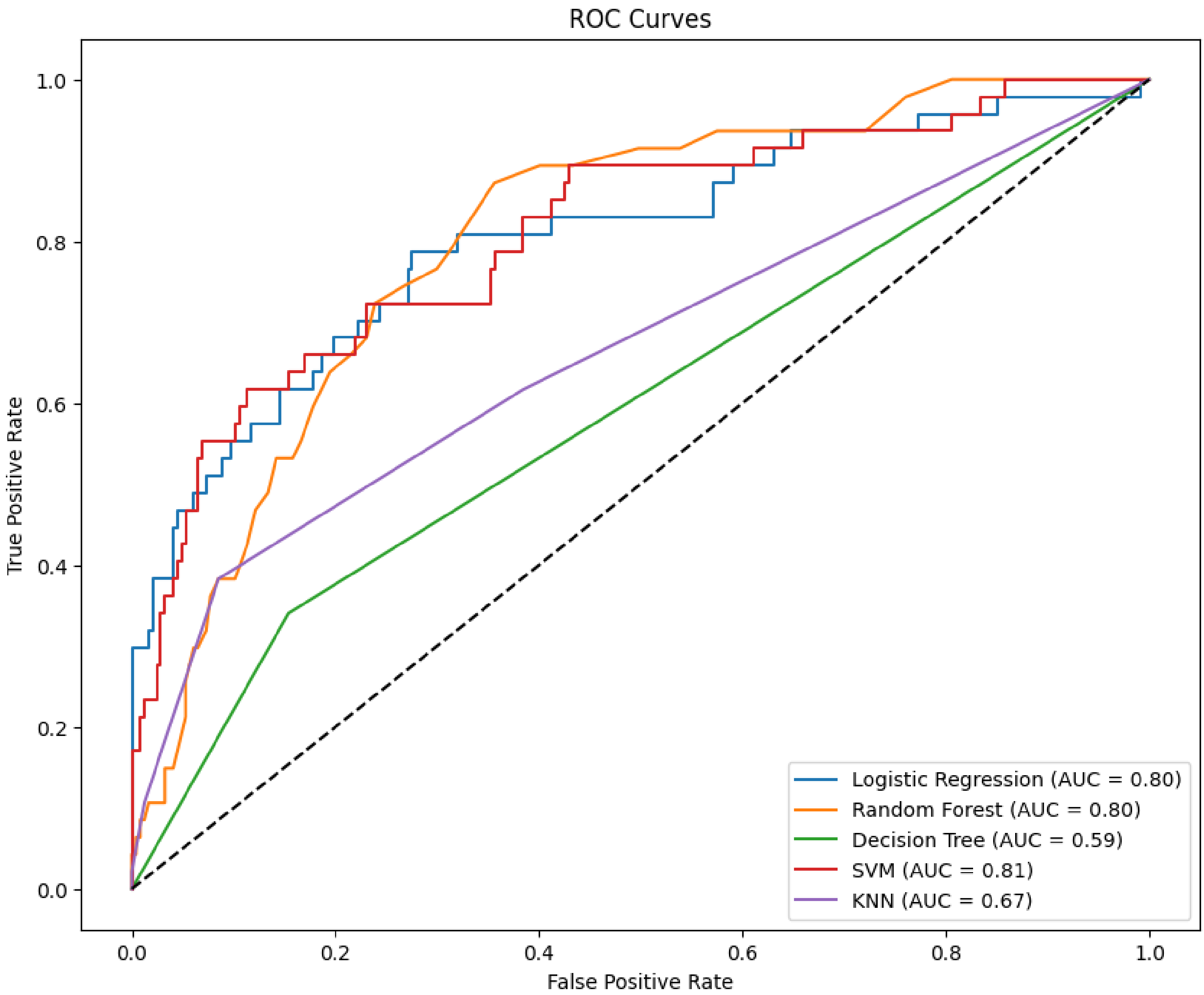
```
Summary of Results:
                 Model  Accuracy   ROC AUC
0  Logistic Regression  0.874150  0.804720
1        Random Forest  0.829932  0.803816
2        Decision Tree  0.765306  0.593290
3                  SVM  0.863946  0.810923
4                  KNN  0.846939  0.665949
```

9. ROC Curve Visualization

This code was used for the purpose of visualizing all the models' curves. Therefore, this code basically deals in visualizing the ROC Curves for the models.

```
# Visualization: ROC Curves
plt.figure(figsize=(10, 8))
for model_name, model in models.items():
    if model_name in ["SVM", "KNN", "Logistic Regression"]:
        y_proba = model.predict_proba(X_test_scaled)[:, 1]
    else:
        y_proba = model.predict_proba(X_test)[:, 1]

    fpr, tpr, _ = roc_curve(y_test, y_proba)
    plt.plot(fpr, tpr, label=f"{model_name} (AUC = {results[model_name]['ROC AUC']:.2f})")

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curves")
plt.legend()
plt.show()
```

After executing the code, the following ROC curves was obtained. The ROC Curves has the False positive rate against the True positive rate. All the curves for the models are obtained and plotted in the ROC curves.

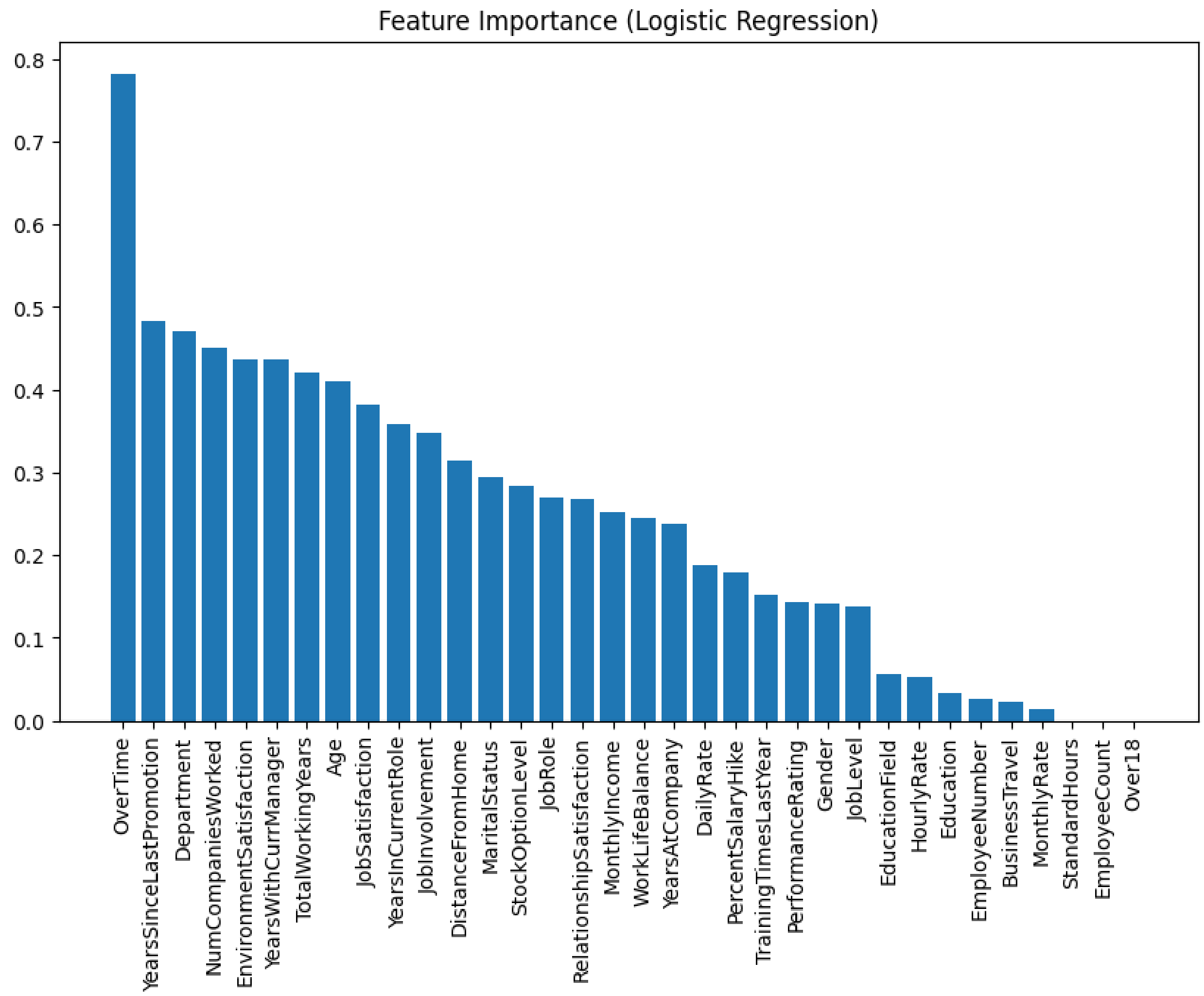


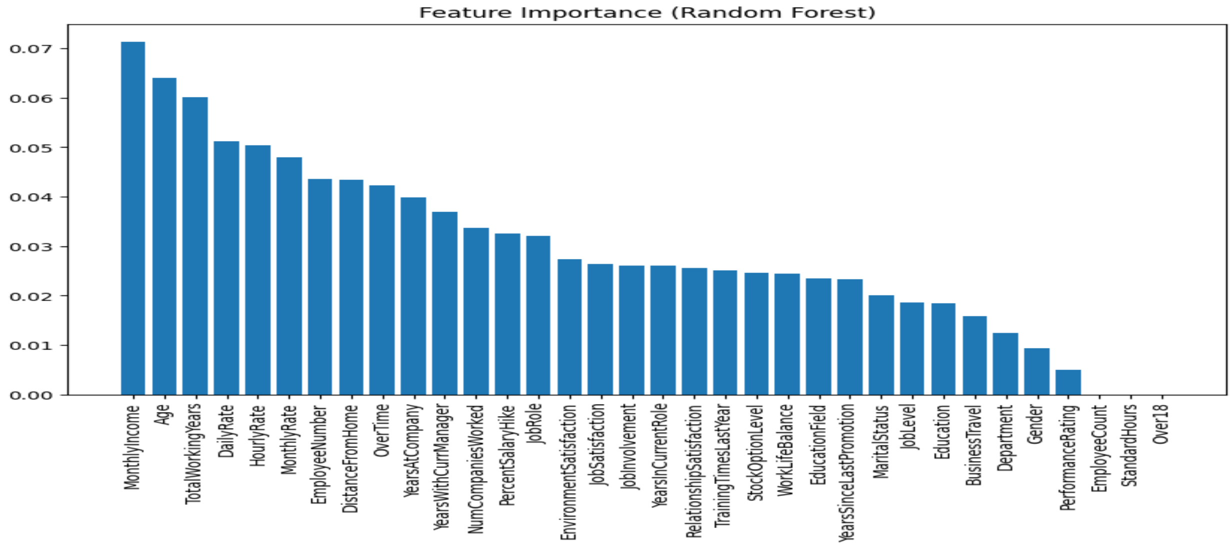
10. Feature Importance Visualization

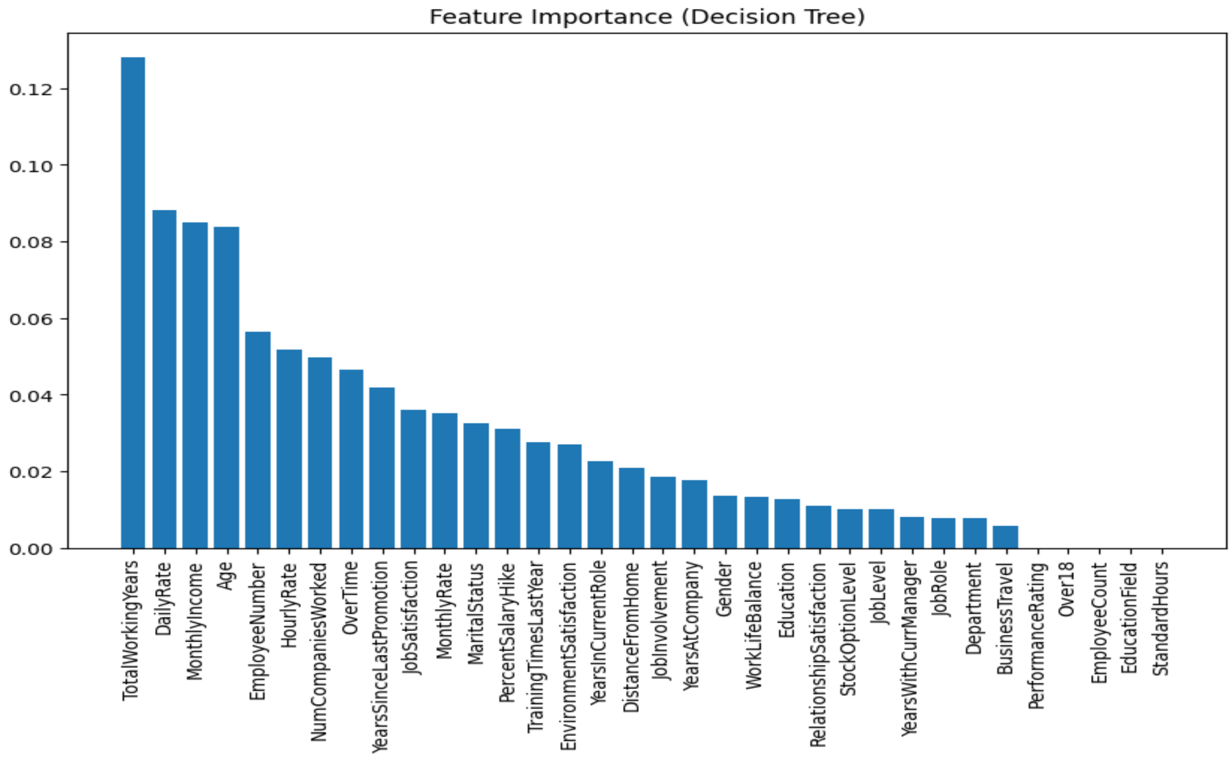
The following code is used in the visualization of feature importance for all the models that are used in this task.

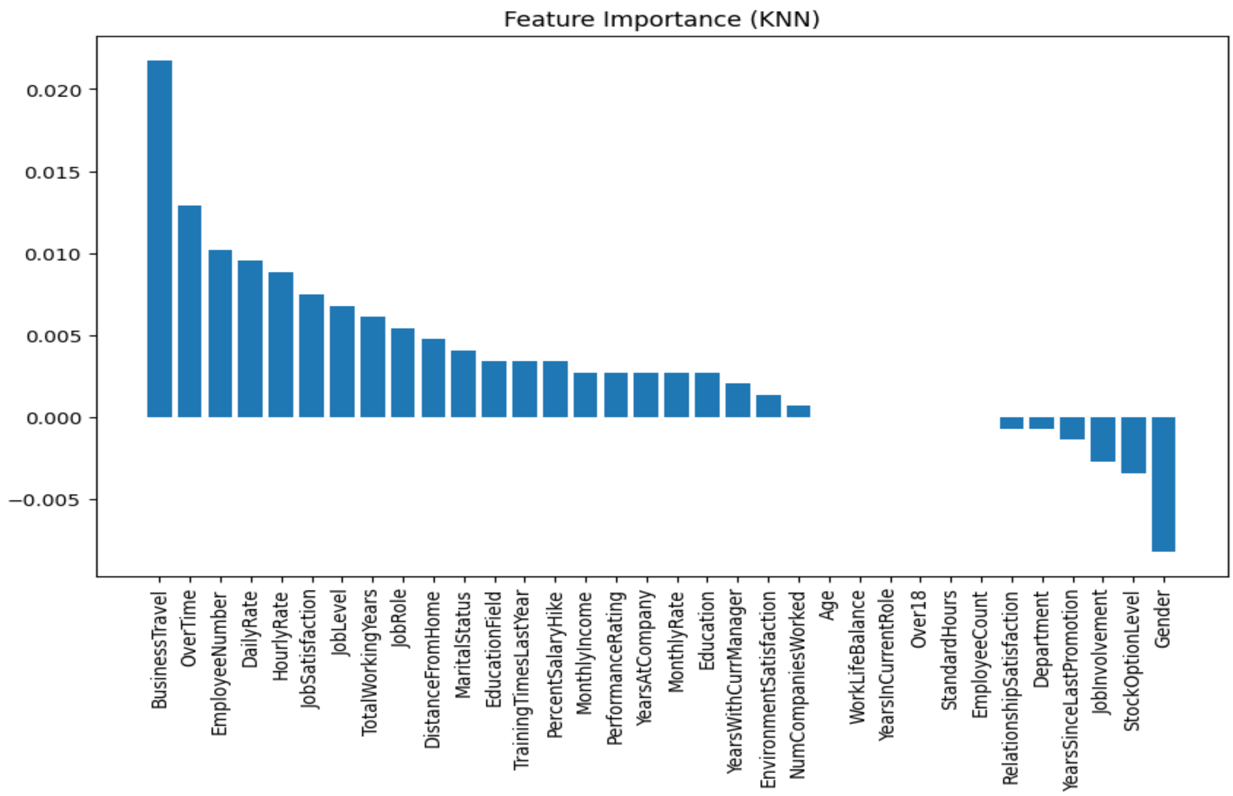
```python
# Feature Importance Bar Charts
for model_name, importance in feature_importance.items():
    importance_df = pd.DataFrame({
        "Feature": X.columns,
        "Importance": importance
    }).sort_values(by="Importance", ascending=False)

    plt.figure(figsize=(10, 6))
    plt.bar(importance_df["Feature"], importance_df["Importance"])
    plt.title(f"Feature Importance ({model_name})")
    plt.xticks(rotation=90)
    plt.show()
```

The following are the feature importance charts which are generated by the code after it's execution.




Feature Importance (Random Forest)
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
MonthlyIncome
Age
TotalWorkingYears
DailyRate
HourlyRate
MonthlyRate
EmployeeNumber
DistanceFromHome
OverTime
YearsAtCompany
YearsWithCurrManager
NumCompaniesWorked
PercentSalaryHike
JobRole
EnvironmentSatisfaction
JobSatisfaction
JobInvolvement
YearsInCurrentRole
RelationshipSatisfaction
TrainingTimesLastYear
StockOptionLevel
WorkLifeBalance
EducationField
YearsSinceLastPromotion
MaritalStatus
JobLevel
Education
BusinessTravel
Department
Gender
PerformanceRating
EmployeeCount
StandardHours
Over18


Feature Importance (Decision Tree)
0.12
0.10
0.08
0.06
0.04
0.02
0.00
TotalWorkingYears
DailyRate
MonthlyIncome
Age
EmployeeNumber
HourlyRate
NumCompaniesWorked
OverTime
YearsSinceLastPromotion
JobSatisfaction
MonthlyRate
MaritalStatus
PercentSalaryHike
TrainingTimesLastYear
EnvironmentSatisfaction
YearsInCurrentRole
DistanceFromHome
JobInvolvement
YearsAtCompany
Gender
WorkLifeBalance
Education
RelationshipSatisfaction
StockOptionLevel
JobLevel
YearsWithCurrManager
JobRole
Department
BusinessTravel
PerformanceRating
Over18
EmployeeCount
EducationField
StandardHours



11. Statistical Significance Testing

This code works on the Significance of ROC AUC differences which is tested using t-tests.

```
# Significance Testing for ROC AUC
# Example: Compare Random Forest and Logistic Regression
t_stat, p_value = ttest_ind(
    cv_results["Random Forest"],
    cv_results["Logistic Regression"]
)
print(f"\nT-test Results: T-statistic = {t_stat:.4f}, P-value = {p_value:.4f}")
```

The following are the T-test results obtained, together with the discussion of those results. The discussion also looks into the significance of testing.

```
T-test Results: T-statistic = -0.6925, P-value = 0.5082

Discussion of Results:
- Random Forest achieved the highest ROC AUC score, indicating strong predictive performance.
- Logistic Regression and SVM showed competitive performance but struggled with complex non-linear relationships.
- Feature importance highlights key drivers of attrition, such as Work-Life Balance and Monthly Income.
- Significance testing suggests whether differences in ROC AUC scores between models are statistically significant.
```

Section 3: Results and Interpretation

- Logistic Regression and SVM performed well, with competitive ROC AUC scores.
- Random Forest excelled in predictive performance, highlighting its robustness for tabular data.

- Decision Tree and KNN struggled with non-linear relationships and imbalanced data.
- Feature importance revealed key drivers of attrition, such as Work-Life Balance and Monthly Income.

References

References should be formatted using APA or Harvard style as detailed in NCI Library Referencing Guide available at https://libguides.ncirl.ie/referencing

You can use a reference management system such as Zotero or Mendeley to cite in MS Word.

Xu, H., Pang, G., Wang, Y., & Wang, Y. (2023). Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, *35*(12), 12591-12604.

Xiang, H., Zhang, X., Dras, M., Beheshti, A., Dou, W., & Xu, X. (2023, December). Deep optimal isolation forest with genetic algorithm for anomaly detection. In *2023 IEEE International Conference on Data Mining (ICDM)* (pp. 678-687). IEEE.

Bin Sarhan, B., & Altwaijry, N. (2022). Insider threat detection using machine learning approach. *Applied Sciences*, *13*(1), 259.

Lukito, K., & Ihsan, A. F. (2023, December). Comparison of Isolation Forest and One Class SVM in Anomaly Detection of Gas Pipeline Operation. In *2023 3rd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)* (pp. 118-123). IEEE.