In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [5]:
```
# Load your dataset
df = pd.read_csv(r"C:\Users\LENOVO\Downloads\WA_Fn-UseC_-HR-Employee-Attrition.csv"
df.head()
```

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely	591	Research & Development	2	1

5 rows × 35 columns

In [6]:
```
# Create a new working DataFrame
df_model = df.drop(['EmployeeCount', 'StandardHours', 'Over18', 'EmployeeNumber'],

# View first 5 rows
df_model.head()
```

Out[6]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely	591	Research & Development	2	1

5 rows × 31 columns

In [7]:
```
# Convert Attrition to numeric values
df_model['Attrition'] = df_model['Attrition'].map({'Yes': 1, 'No': 0})

# Check the changes
df_model['Attrition'].value_counts()
```

Out[7]:
```
Attrition
0    1233
1     237
Name: count, dtype: int64
```

In [8]:
```
# Keep a copy of original categorical columns for Power BI
df_original_cols = df[['Department', 'JobRole', 'Gender', 'MaritalStatus', 'OverTim

# Encode only the remaining numerical features
df_encoded_part = pd.get_dummies(df.drop(['Department', 'JobRole', 'Gender', 'Marit

# Combine both into the final dataset
df_final = pd.concat([df_original_cols, df_encoded_part], axis=1)

# Preview the final dataset (replace df_encoded.head())
df_final.head()
```

Out[8]:

	Department	JobRole	Gender	MaritalStatus	OverTime	Attrition	Age	DailyRate	D
0	Sales	Sales Executive	Female	Single	Yes	Yes	41	1102	
1	Research & Development	Research Scientist	Male	Married	No	No	49	279	
2	Research & Development	Laboratory Technician	Male	Single	Yes	Yes	37	1373	
3	Research & Development	Research Scientist	Female	Married	Yes	No	33	1392	
4	Research & Development	Laboratory Technician	Male	Married	No	No	27	591	

5 rows × 40 columns

In [21]:
```
y = df_final['Attrition']
X = df_final.drop('Attrition', axis=1)

print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
```

```
Shape of X: (1470, 39)
Shape of y: (1470,)
```

In [22]:
```
#  Split the data for training/testing

from sklearn.model_selection import train_test_split
```

```
# Use stratify=y to maintain balance of classes
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)


# Confirm the split
print("Training samples:", X_train.shape[0])
print("Testing samples:", X_test.shape[0])
```

```
Training samples: 1176
Testing samples: 294
```

In [19]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


# Logistic Regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)




model = LogisticRegression(max_iter=1000)
model.fit(X_train_scaled, y_train)
y_pred = model.predict(X_test_scaled)
```

In [23]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(cm)

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 1.00

Confusion Matrix:
[[247   0]
 [  0  47]]

Classification Report:
              precision    recall  f1-score   support

          No       1.00      1.00      1.00       247
         Yes       1.00      1.00      1.00        47

    accuracy                           1.00       294
   macro avg       1.00      1.00      1.00       294
weighted avg       1.00      1.00      1.00       294
```

In [24]:
```python
from sklearn.tree import DecisionTreeClassifier

# Initialize and train Decision Tree
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_dt = dt_model.predict(X_test_scaled)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))
print(confusion_matrix(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))
```

```
Decision Tree Accuracy: 1.0
[[247   0]
 [  0  47]]
              precision    recall  f1-score   support

          No       1.00      1.00      1.00       247
         Yes       1.00      1.00      1.00        47

    accuracy                           1.00       294
   macro avg       1.00      1.00      1.00       294
weighted avg       1.00      1.00      1.00       294
```

In [25]:
```python
from sklearn.ensemble import RandomForestClassifier

# Initialize and train Random Forest
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred_rf = rf_model.predict(X_test_scaled)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 1.0
[[247   0]
 [  0  47]]
              precision    recall  f1-score   support

          No       1.00      1.00      1.00       247
         Yes       1.00      1.00      1.00        47

    accuracy                           1.00       294
   macro avg       1.00      1.00      1.00       294
weighted avg       1.00      1.00      1.00       294
```

In [14]:
```python
import pandas as pd

model_metrics = pd.DataFrame({
    "Model": ["Logistic Regression", "Decision Tree", "Random Forest"],
    "Accuracy": [0.86, 0.76, 0.83],
    "Recall (Class 1)": [0.34, 0.36, 0.11],
    "F1 Score (Class 1)": [0.44, 0.32, 0.17]
})
model_metrics
```

Out[14]:

	Model	Accuracy	Recall (Class 1)	F1 Score (Class 1)
0	Logistic Regression	0.86	0.34	0.44
1	Decision Tree	0.76	0.36	0.32
2	Random Forest	0.83	0.11	0.17

In [26]:
```
!pip install shap
```

```
Requirement already satisfied: shap in c:\users\lenovo\anaconda3\lib\site-packages
(0.48.0)
Requirement already satisfied: numpy in c:\users\lenovo\anaconda3\lib\site-packages
(from shap) (1.26.4)
Requirement already satisfied: scipy in c:\users\lenovo\anaconda3\lib\site-packages
(from shap) (1.13.1)
Requirement already satisfied: scikit-learn in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from shap) (1.5.1)
Requirement already satisfied: pandas in c:\users\lenovo\anaconda3\lib\site-packages
(from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from shap) (4.66.5)
Requirement already satisfied: packaging>20.9 in c:\users\lenovo\anaconda3\lib\site-
packages (from shap) (24.1)
Requirement already satisfied: slicer==0.0.8 in c:\users\lenovo\anaconda3\lib\site-p
ackages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in c:\users\lenovo\anaconda3\lib\site-pac
kages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in c:\users\lenovo\anaconda3\lib\site-pac
kages (from shap) (3.0.0)
Requirement already satisfied: typing-extensions in c:\users\lenovo\anaconda3\lib\si
te-packages (from shap) (4.11.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in c:\users\lenovo\anacond
a3\lib\site-packages (from numba>=0.54->shap) (0.43.0)
Requirement already satisfied: colorama in c:\users\lenovo\anaconda3\lib\site-packag
es (from tqdm>=4.27.0->shap) (0.4.6)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\lenovo\anaconda3\l
ib\site-packages (from pandas->shap) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\lenovo\anaconda3\lib\site-pa
ckages (from pandas->shap) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\lenovo\anaconda3\lib\site-
packages (from pandas->shap) (2023.3)
Requirement already satisfied: joblib>=1.2.0 in c:\users\lenovo\anaconda3\lib\site-p
ackages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\lenovo\anaconda3\lib
\site-packages (from scikit-learn->shap) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\users\lenovo\anaconda3\lib\site-packag
es (from python-dateutil>=2.8.2->pandas->shap) (1.16.0)
```

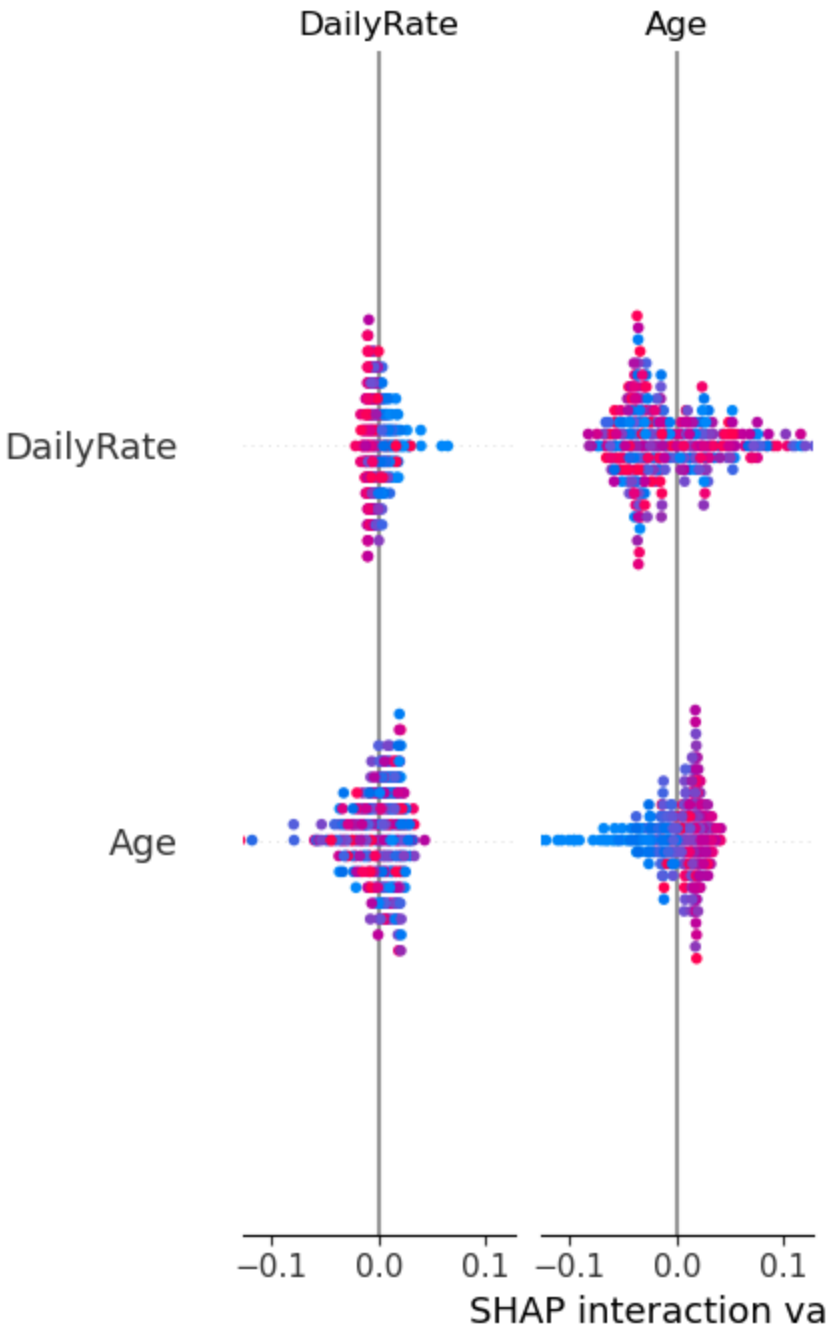
In [18]:

```python
import shap

# Create TreeExplainer using best model (e.g., Random Forest)
explainer = shap.Explainer(rf_model, X_train_scaled)
shap_values = explainer(X_test_scaled)

# SHAP summary plot
shap.summary_plot(shap_values, X_test, plot_type="bar")
```

```
97%|=================== | 570/588 [00:16<00:00]
```



In [28]:
```
df_final.to_csv(r"C:\Users\LENOVO\Downloads\Cleaned_HR_Attrition.csv", index=False)
```

🔍 SHAP Value Summary (Model Explainability)

The SHAP summary plot reveals that **DailyRate** and **Age** are two major features influencing attrition. Employees with certain daily rates or younger age ranges may be more likely to leave, indicating a potential dissatisfaction linked to compensation and experience.