capstone-1

July 29, 2024

Development of Machine Learning Models:

With a clean and well-structured dataset, I developed various machine-learning models to forecast health trends and outcomes. This phase required a deep dive into predictive analytics, leveraging my expertise in Python and popular ML libraries such as Scikit-learn and TensorFlow.

The process began with exploratory data analysis to identify patterns, correlations, and potential predictive features within the health data. Based on these insights, I have developed and am still developing a range of ML models, including: 1. Classification Models: To identify high-risk populations or areas. Examples include Logistic Regression and Decision Trees. 2. Regression Models: To understand the relationships between various health indicators. Examples include Linear Regression and Random Forests. 3. Clustering Algorithms: To group similar health outcomes or demographic profiles. Examples include K-Means Clustering.

Each model was carefully tuned and validated using cross-validation techniques to ensure accuracy and generalizability. I paid particular attention to model interpretability, recognizing the importance of transparency in public health decision-making.

I am also in the process of integrating these ML models with Power BI. This involves developing custom Python scripts that can be executed within the Power BI environment, allowing for real-time predictions and dynamic dashboard updates based on the latest data. This integration requires careful consideration of data flow, processing speed, and resource utilization to ensure smooth dashboard performance.

#IMPORTING THE DTA

[1]:
```
import pandas as pd

# Load the dataset
file_path = '3011_Rpt_AgencyDoc_Detail Self-sufficiency survey.xlsx'
df = pd.read_excel(file_path)

# Display the first few rows of the dataset
print(df.head())

# Display the column names
print(df.columns)
```

```
   Client ID  Current Age          County  State  Zip Code  \
0      16676           78        RANDOLPH     MO     65270
1      17476           33  RANDOLPH COUNTY     MO     65270
```

```
2      16676           78         RANDOLPH    MO     65270
3      17475           46  RANDOLPH COUNTY    MO     65270
4      17478           67  RANDOLPH COUNTY    MO     65270

                        Race         Ethnicity  Gender Document Date  \
0  Black or African American               NaN    Male    2018-02-12
1                      White               NaN  Female    2018-03-06
2  Black or African American               NaN    Male    2018-03-07
3                      White  Central American  Female    2018-03-09
4                      White               NaN  Female    2018-03-12

  Document Type  …  \
0     Admission  …
1     Admission  …
2     Admission  …
3     Admission  …
4     Admission  …

  Can the client meet some basic living needs without any assistance?  \
0  a. Unable to meet basic needs such as hygiene,…
1  b. Can meet a few but not all needs of daily l…
2  c. Can meet most but not all daily living need…
3  e. Able to provide beyond basic needs of daily…
4  c. Can meet most but not all daily living need…

  Does the client have mild or no mental health issues?  \
0  a. Danger to self or others; recurring suicida…
1  c. Mild symptoms may be present but are transi…
2  a. Danger to self or others; recurring suicida…
3  e. Symptoms are absent or rare; good or superi…
4  e. Symptoms are absent or rare; good or superi…

  Is the client seriously dependent on alcohol or drugs?  \
0  a. Meets criteria for severe abuse/dependence;…
1      e. No drug/alcohol abuse in last 6 months [5]
2  a. Meets criteria for severe abuse/dependence;…
3      e. No drug/alcohol abuse in last 6 months [5]
4      e. No drug/alcohol abuse in last 6 months [5]

  Does the client receive any support from family/friends?  \
0  a. Lack of necessary support from family or fr…
1  e. Has healthy/expanding support network; hous…
2  b. Family/friends may be supportive but lack a…
3  d. Strong support from family or friends; hous…
4  e. Has healthy/expanding support network; hous…

  Does the client have access to any means of transportation?  \
0  a. No access to transportation, public or priv…
```

```
1  e. Transportation is readily available and aff…
2  a. No access to transportation, public or priv…
3  e. Transportation is readily available and aff…
4  d. Transportation (including bus) is generally…

                       Is the client in crisis mode?  \
0  b. Socially isolated and/or no social skills a…
1  c. Lacks knowledge of ways to become involved …
2  b. Socially isolated and/or no social skills a…
3  d. Some community involvement (church, advisor…
4  b. Socially isolated and/or no social skills a…

                  Is the client's safety threatened?  \
0  a. Home/residence is not safe, lethality is hi…
1          e. Home is apparently safe and stable [5]
2  a. Home/residence is not safe, lethality is hi…
3          e. Home is apparently safe and stable [5]
4          e. Home is apparently safe and stable [5]

              Is the client's family in crisis mode?  \
0  a. Parenting skill are lacking and there is no…
1               d. Parenting skills are adequate [4]
2  a. Parenting skill are lacking and there is no…
3         e. Parenting skills are well developed [5]
4         e. Parenting skills are well developed [5]

  Is the client facing any bankruptcy/foreclosure/eviction issues?  \
0                      a. Bankruptcy/Foreclosure [1]
1                    c. Has a credit repair plan [3]
2                       b. Outstanding judgments [2]
3         e. Good Credit / Manageable Debt Ratio [5]
4                    c. Has a credit repair plan [3]

  Does the client have any acute or chronic conditions that are impacting all
aspects of their life?
0  a. In crisis - acute or chronic symptoms affec…
1  a. In crisis - acute or chronic symptoms affec…
2  a. In crisis - acute or chronic symptoms affec…
3         e. Thriving - no identified disability [5]
4  b. Vulnerable- sometimes or periodic has acute…

[5 rows x 29 columns]
Index(['Client ID', 'Current Age', 'County  ', 'State', 'Zip Code', 'Race',
       'Ethnicity', 'Gender', 'Document Date', 'Document Type',
       'Does the client get adequate income?', 'Is the client employed?',
       'Does the client have housing and is not at any immediate risk of losing
the housing?',
       'Does the client have the means to acquire food and prepare it?',
       'Does the client have access to any form of child care? (Skip if no
children of child care age)',
       'Is one or more of the eligible children enrolled in a school? (Skip if
no children of school age)',
       'Does the client have a high school diploma or a GED?',
       'Does the client have any unresolved legal issues in the past 12
months?',
       'Does the client have medical insurance coverage?',
       'Can the client meet some basic living needs without any assistance?',
       'Does the client have mild or no mental health issues?',
       'Is the client seriously dependent on alcohol or drugs?',
       'Does the client receive any support from family/friends?',
       'Does the client have access to any means of transportation?',
       'Is the client in crisis mode?', 'Is the client's safety threatened?',
       'Is the client's family in crisis mode?',
       'Is the client facing any bankruptcy/foreclosure/eviction issues?',
       'Does the client have any acute or chronic conditions that are impacting
all aspects of their life?'],
      dtype='object')
```

#DATA CLEANING

[2]:
```python
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer

# Fill missing values with the mean for numerical columns
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
if len(numerical_cols) > 0:
    imputer = SimpleImputer(strategy='mean')
    df[numerical_cols] = imputer.fit_transform(df[numerical_cols])

# Encode categorical variables
categorical_cols = df.select_dtypes(include=['object']).columns
label_encoders = {}
for column in categorical_cols:
    label_encoders[column] = LabelEncoder()
    df[column] = label_encoders[column].fit_transform(df[column])

# Display the cleaned dataset
print(df.head())
```

```
   Client ID  Current Age  County    State  Zip Code  Race  Ethnicity  Gender  \
0    16676.0         78.0       23       3   65270.0     5          8       1
1    17476.0         33.0       24       3   65270.0    15          8       0
2    16676.0         78.0       23       3   65270.0     5          8       1
3    17475.0         46.0       24       3   65270.0    15          1       0
4    17478.0         67.0       24       3   65270.0    15          8       0
```

```
  Document Date  Document Type  …  \
0    2018-02-12              0  …   
1    2018-03-06              0  …   
2    2018-03-07              0  …   
3    2018-03-09              0  …   
4    2018-03-12              0  …   

   Can the client meet some basic living needs without any assistance?  \
0                                                  0                     
1                                                  1                     
2                                                  2                     
3                                                  4                     
4                                                  2                     

   Does the client have mild or no mental health issues?  \
0                                                  0       
1                                                  2       
2                                                  0       
3                                                  4       
4                                                  4       

   Is the client seriously dependent on alcohol or drugs?  \
0                                                  0        
1                                                  4        
2                                                  0        
3                                                  4        
4                                                  4        

   Does the client receive any support from family/friends?  \
0                                                  0          
1                                                  4          
2                                                  1          
3                                                  3          
4                                                  4          

   Does the client have access to any means of transportation?  \
0                                                  0             
1                                                  4             
2                                                  0             
3                                                  4             
4                                                  3             

   Is the client in crisis mode?  Is the client’s safety threatened?  \
0                              1                                   0   
1                              2                                   4   
2                              1                                   0   
3                              3                                   4   
4                              1                                   4   
```

```
   Is the client's family in crisis mode?  \
0                                        0
1                                        3
2                                        0
3                                        4
4                                        4

   Is the client facing any bankruptcy/foreclosure/eviction issues?  \
0                                                  0
1                                                  2
2                                                  1
3                                                  4
4                                                  2

   Does the client have any acute or chronic conditions that are impacting all
aspects of their life?
0                                                  0
1                                                  0
2                                                  0
3                                                  4
4                                                  1

[5 rows x 29 columns]
```

#SPLITTING THE DATA

[3]:
```
from sklearn.model_selection import train_test_split

# Define the features and the target variable
target_column = 'Is the client employed?'  # Replace with your chosen target␣
 ↪column
X = df.drop([target_column, 'Document Date'], axis=1)  # Exclude the 'Document␣
 ↪Date' column
y = df[target_column]

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

FEATURE SCALING

[4]:
```
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

#CLASSIFICATION MODELS

```
[5]: from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

# Logistic Regression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)
print("Logistic Regression:\n", classification_report(y_test, y_pred_log_reg))

# Decision Tree
dec_tree = DecisionTreeClassifier()
dec_tree.fit(X_train, y_train)
y_pred_dec_tree = dec_tree.predict(X_test)
print("Decision Tree:\n", classification_report(y_test, y_pred_dec_tree))

# Random Forest
rand_forest = RandomForestClassifier()
rand_forest.fit(X_train, y_train)
y_pred_rand_forest = rand_forest.predict(X_test)
print("Random Forest:\n", classification_report(y_test, y_pred_rand_forest))

# Gradient Boosting
grad_boost = GradientBoostingClassifier()
grad_boost.fit(X_train, y_train)
y_pred_grad_boost = grad_boost.predict(X_test)
print("Gradient Boosting:\n", classification_report(y_test, y_pred_grad_boost))

# Support Vector Machine
svc = SVC()
svc.fit(X_train, y_train)
y_pred_svc = svc.predict(X_test)
print("Support Vector Machine:\n", classification_report(y_test, y_pred_svc))

# k-Nearest Neighbors
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)
print("k-Nearest Neighbors:\n", classification_report(y_test, y_pred_knn))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:460:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

Logistic Regression:
               precision    recall  f1-score   support

           0       0.71      0.92      0.80       351
           1       0.17      0.01      0.02        91
           2       0.50      0.56      0.53       100
           3       0.33      0.10      0.15        30
           4       0.59      0.36      0.44        28
           5       0.00      0.00      0.00         2
    accuracy                           0.65       602
   macro avg       0.38      0.33      0.32       602
weighted avg       0.56      0.65      0.59       602

Decision Tree:
              precision    recall  f1-score   support

           0       0.82      0.83      0.83       351
           1       0.33      0.32      0.33        91
           2       0.44      0.46      0.45       100
           3       0.21      0.27      0.23        30
           4       0.40      0.29      0.33        28
           5       0.00      0.00      0.00         2

    accuracy                           0.63       602
   macro avg       0.37      0.36      0.36       602
weighted avg       0.63      0.63      0.63       602


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

Random Forest:
              precision    recall  f1-score   support

           0       0.78      0.95      0.86       351
           1       0.53      0.26      0.35        91
           2       0.66      0.69      0.67       100
           3       0.64      0.23      0.34        30
           4       0.75      0.32      0.45        28
           5       0.00      0.00      0.00         2

    accuracy                           0.74       602
   macro avg       0.56      0.41      0.45       602
weighted avg       0.71      0.74      0.70       602
```

```
Gradient Boosting:
              precision    recall  f1-score   support

           0       0.84      0.92      0.88       351
           1       0.52      0.40      0.45        91
           2       0.59      0.66      0.62       100
           3       0.25      0.17      0.20        30
           4       0.82      0.32      0.46        28
           5       0.00      0.00      0.00         2

    accuracy                           0.73       602
   macro avg       0.50      0.41      0.44       602
weighted avg       0.72      0.73      0.71       602

Support Vector Machine:
              precision    recall  f1-score   support

           0       0.73      0.97      0.83       351
           1       0.50      0.07      0.12        91
           2       0.56      0.64      0.60       100
           3       0.50      0.03      0.06        30
           4       0.80      0.29      0.42        28
           5       0.00      0.00      0.00         2

    accuracy                           0.70       602
   macro avg       0.52      0.33      0.34       602
weighted avg       0.66      0.70      0.63       602

k-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.74      0.93      0.82       351
           1       0.41      0.16      0.23        91
           2       0.55      0.56      0.56       100
           3       0.40      0.13      0.20        30
           4       0.75      0.32      0.45        28
           5       0.00      0.00      0.00         2

    accuracy                           0.68       602
   macro avg       0.47      0.35      0.38       602
weighted avg       0.64      0.68      0.64       602

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

#REGRESSION MODELS

```
[6]: from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Define the features and the target variable for regression
target_column_regression = 'Current Age'  # Replace with your actual continuous␣
 ↪target column name
X = df.drop([target_column_regression, 'Document Date'], axis=1)  # Exclude the␣
 ↪'Document Date' column
y = df[target_column_regression]

# Split the dataset into training and test sets again for regression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin_reg = lin_reg.predict(X_test)
```

```
print("Linear Regression:\n", mean_squared_error(y_test, y_pred_lin_reg),␣
 ↪r2_score(y_test, y_pred_lin_reg))

# Ridge Regression
ridge_reg = Ridge()
ridge_reg.fit(X_train, y_train)
y_pred_ridge_reg = ridge_reg.predict(X_test)
print("Ridge Regression:\n", mean_squared_error(y_test, y_pred_ridge_reg),␣
 ↪r2_score(y_test, y_pred_ridge_reg))

# Lasso Regression
lasso_reg = Lasso()
lasso_reg.fit(X_train, y_train)
y_pred_lasso_reg = lasso_reg.predict(X_test)
print("Lasso Regression:\n", mean_squared_error(y_test, y_pred_lasso_reg),␣
 ↪r2_score(y_test, y_pred_lasso_reg))

# Decision Tree Regressor
dec_tree_reg = DecisionTreeRegressor()
dec_tree_reg.fit(X_train, y_train)
y_pred_dec_tree_reg = dec_tree_reg.predict(X_test)
print("Decision Tree Regressor:\n", mean_squared_error(y_test,␣
 ↪y_pred_dec_tree_reg), r2_score(y_test, y_pred_dec_tree_reg))

# Random Forest Regressor
rand_forest_reg = RandomForestRegressor()
rand_forest_reg.fit(X_train, y_train)
y_pred_rand_forest_reg = rand_forest_reg.predict(X_test)
print("Random Forest Regressor:\n", mean_squared_error(y_test,␣
 ↪y_pred_rand_forest_reg), r2_score(y_test, y_pred_rand_forest_reg))

# Gradient Boosting Regressor
grad_boost_reg = GradientBoostingRegressor()
grad_boost_reg.fit(X_train, y_train)
y_pred_grad_boost_reg = grad_boost_reg.predict(X_test)
print("Gradient Boosting Regressor:\n", mean_squared_error(y_test,␣
 ↪y_pred_grad_boost_reg), r2_score(y_test, y_pred_grad_boost_reg))
```

```
Linear Regression:
 148.8049671066826 0.23698381923210265
Ridge Regression:
 148.7172173104601 0.23743376734659571
Lasso Regression:
 158.19556643719656 0.18883234031572071
Decision Tree Regressor:
 150.6312292358804 0.2276194305158138
Random Forest Regressor:
 72.5156976744186 0.6281666414697957
Gradient Boosting Regressor:
 98.90700971761136 0.49284187031337345
```

0.0.1 Explanation of Regression Results

Here are the results of different regression models applied to the dataset:

1. **Linear Regression**:
    - **Mean Squared Error (MSE)**: 148.80
    - **R-squared (R^2)**: 0.237
    - Indicates moderate accuracy in predicting the target variable, with 23.7% of variance explained by the model.
2. **Ridge Regression**:
    - **MSE**: 148.72
    - **R^2**: 0.237
    - Similar performance to linear regression, showing a slight improvement in prediction accuracy.
3. **Lasso Regression**:
    - **MSE**: 158.20
    - **R^2**: 0.189
    - Slightly worse performance compared to linear and ridge regression, with 18.9% of variance explained.
4. **Decision Tree Regressor**:
    - **MSE**: 150.63
    - **R^2**: 0.228
    - Similar performance to linear regression, with 22.8% of variance explained.
5. **Random Forest Regressor**:
    - **MSE**: 72.52
    - **R^2**: 0.628
    - Significantly better performance, explaining 62.8% of the variance, indicating a strong predictive model.
6. **Gradient Boosting Regressor**:
    - **MSE**: 98.91
    - **R^2**: 0.493
    - Good performance, with 49.3% of variance explained, better than linear models but not as strong as random forest.

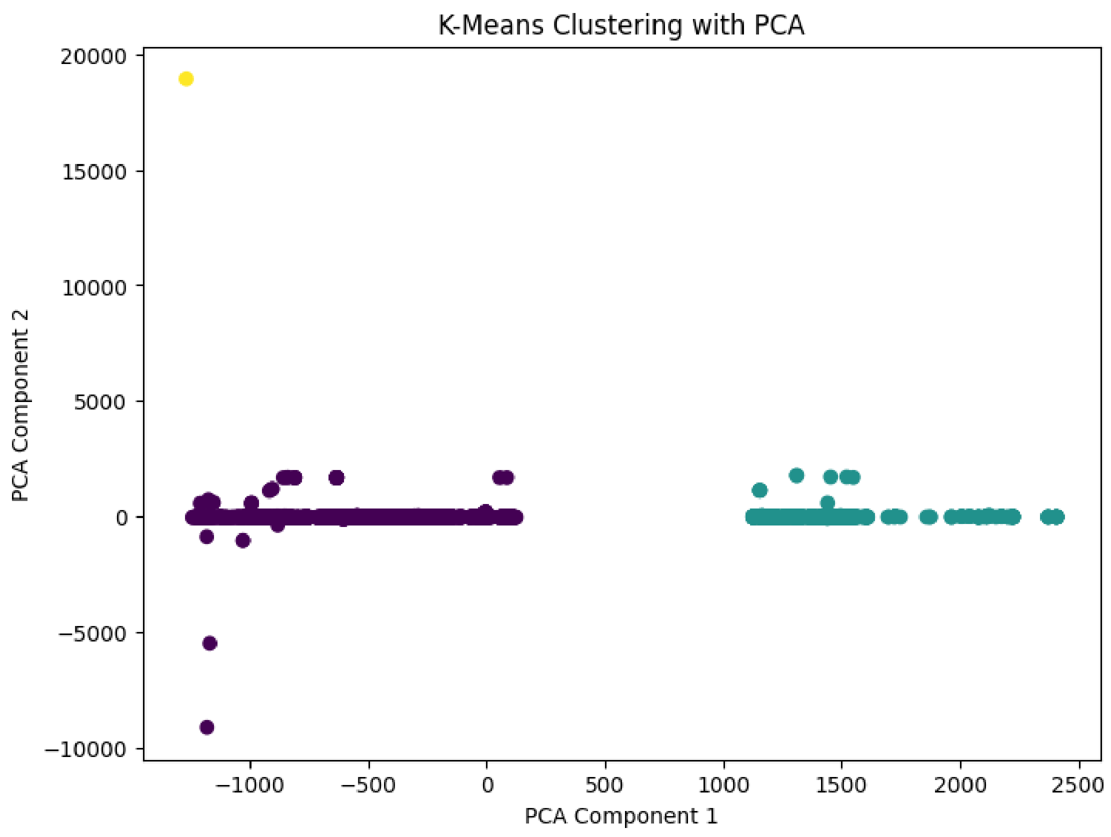
0.0.2 Summary

- **Random Forest Regressor** performs the best with the lowest MSE (72.52) and highest R^2 (0.628), indicating it is the most accurate model for this dataset.
- **Gradient Boosting Regressor** also shows good performance with moderate MSE (98.91) and R^2 (0.493).
- Linear, Ridge, Lasso, and Decision Tree regressors have similar performance, with moderate accuracy and higher MSE compared to ensemble methods.

#CLUSTERING MODELS

[7]:
```python
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# K-Means Clustering
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
df['kmeans_labels'] = kmeans.labels_

# Hierarchical Clustering
agg_clustering = AgglomerativeClustering(n_clusters=3)
df['agg_clustering_labels'] = agg_clustering.fit_predict(X)

# DBSCAN
dbscan = DBSCAN()
df['dbscan_labels'] = dbscan.fit_predict(X)

# Example of Dimensionality Reduction with PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=df['kmeans_labels'])
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.title('K-Means Clustering with PCA')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



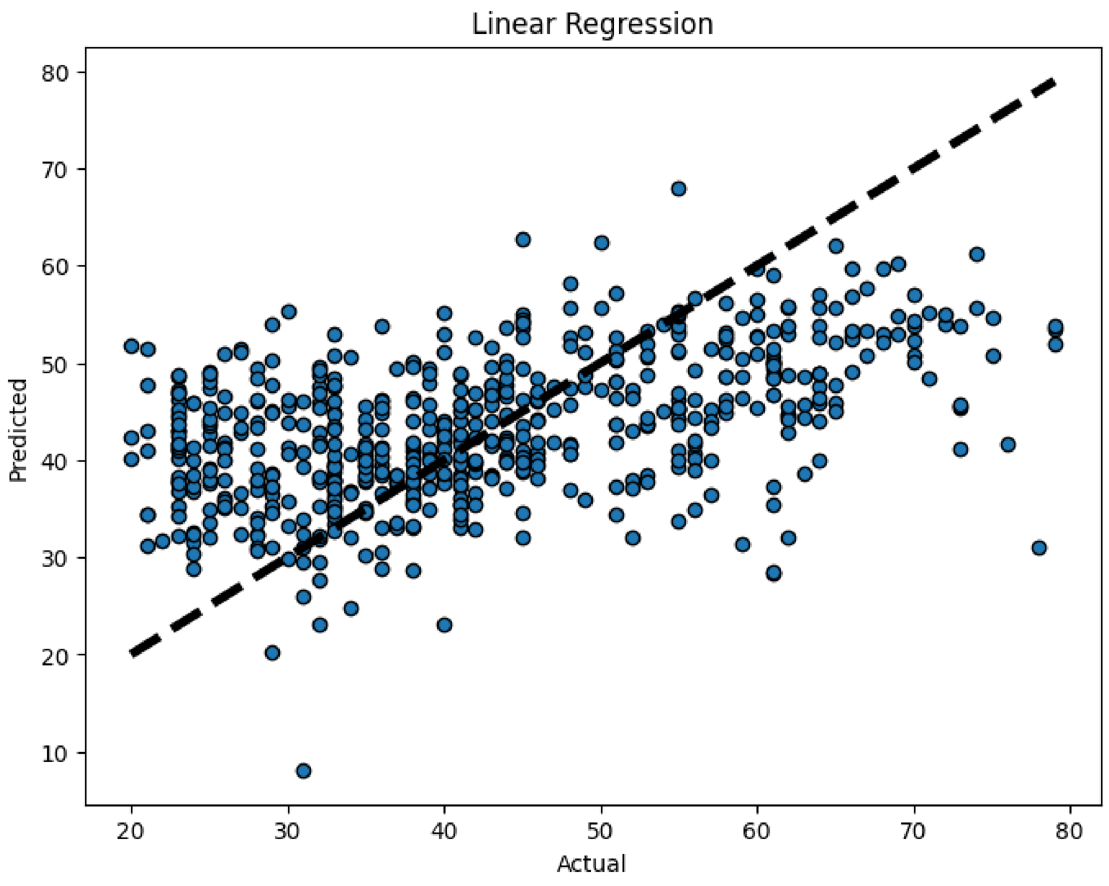
#Regression model for visualization

[8]:
```python
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Function to plot actual vs predicted values
def plot_regression_results(y_true, y_pred, title):
    plt.figure(figsize=(8, 6))
    plt.scatter(y_true, y_pred, edgecolors=(0, 0, 0))
    plt.plot([y_true.min(), y_true.max()], [y_true.min(), y_true.max()], 'k--',␣
 ↪lw=4)
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.show()

# Linear Regression
lin_reg = LinearRegression()
```

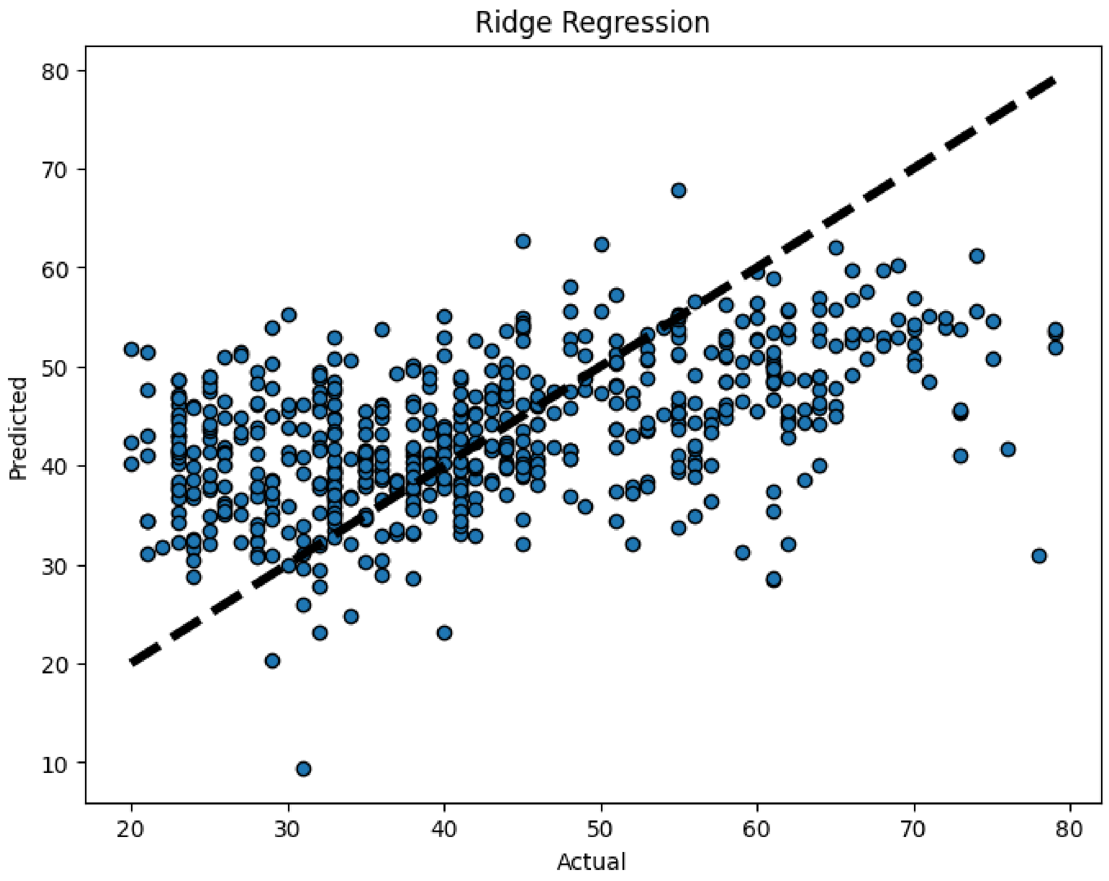
```
lin_reg.fit(X_train, y_train)
y_pred_lin_reg = lin_reg.predict(X_test)
print("Linear Regression:\nMSE:", mean_squared_error(y_test, y_pred_lin_reg),␣
 ↪"\nR2:", r2_score(y_test, y_pred_lin_reg))
plot_regression_results(y_test, y_pred_lin_reg, "Linear Regression")

# Ridge Regression
ridge_reg = Ridge()
ridge_reg.fit(X_train, y_train)
y_pred_ridge_reg = ridge_reg.predict(X_test)
print("Ridge Regression:\nMSE:", mean_squared_error(y_test, y_pred_ridge_reg),␣
 ↪"\nR2:", r2_score(y_test, y_pred_ridge_reg))
plot_regression_results(y_test, y_pred_ridge_reg, "Ridge Regression")

# Lasso Regression
lasso_reg = Lasso()
lasso_reg.fit(X_train, y_train)
y_pred_lasso_reg = lasso_reg.predict(X_test)
print("Lasso Regression:\nMSE:", mean_squared_error(y_test, y_pred_lasso_reg),␣
 ↪"\nR2:", r2_score(y_test, y_pred_lasso_reg))
plot_regression_results(y_test, y_pred_lasso_reg, "Lasso Regression")

# Decision Tree Regressor
dec_tree_reg = DecisionTreeRegressor()
dec_tree_reg.fit(X_train, y_train)
y_pred_dec_tree_reg = dec_tree_reg.predict(X_test)
print("Decision Tree Regressor:\nMSE:", mean_squared_error(y_test,␣
 ↪y_pred_dec_tree_reg), "\nR2:", r2_score(y_test, y_pred_dec_tree_reg))
plot_regression_results(y_test, y_pred_dec_tree_reg, "Decision Tree Regressor")

# Random Forest Regressor
rand_forest_reg = RandomForestRegressor()
rand_forest_reg.fit(X_train, y_train)
y_pred_rand_forest_reg = rand_forest_reg.predict(X_test)
print("Random Forest Regressor:\nMSE:", mean_squared_error(y_test,␣
 ↪y_pred_rand_forest_reg), "\nR2:", r2_score(y_test, y_pred_rand_forest_reg))
plot_regression_results(y_test, y_pred_rand_forest_reg, "Random Forest␣
 ↪Regressor")

# Gradient Boosting Regressor
grad_boost_reg = GradientBoostingRegressor()
grad_boost_reg.fit(X_train, y_train)
y_pred_grad_boost_reg = grad_boost_reg.predict(X_test)
print("Gradient Boosting Regressor:\nMSE:", mean_squared_error(y_test,␣
 ↪y_pred_grad_boost_reg), "\nR2:", r2_score(y_test, y_pred_grad_boost_reg))
```

```
plot_regression_results(y_test, y_pred_grad_boost_reg, "Gradient Boosting␣
↪Regressor")
```

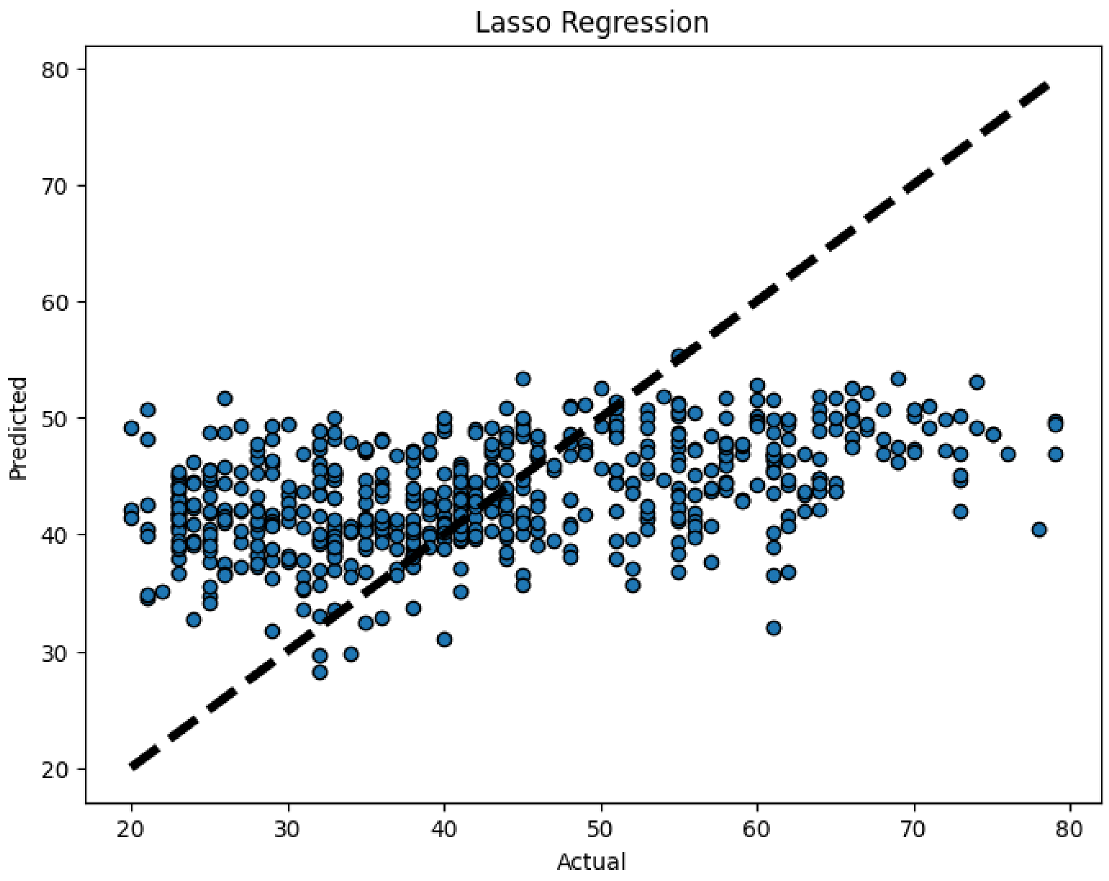
```
Linear Regression:
MSE: 148.8049671066826
R2: 0.23698381923210265
```



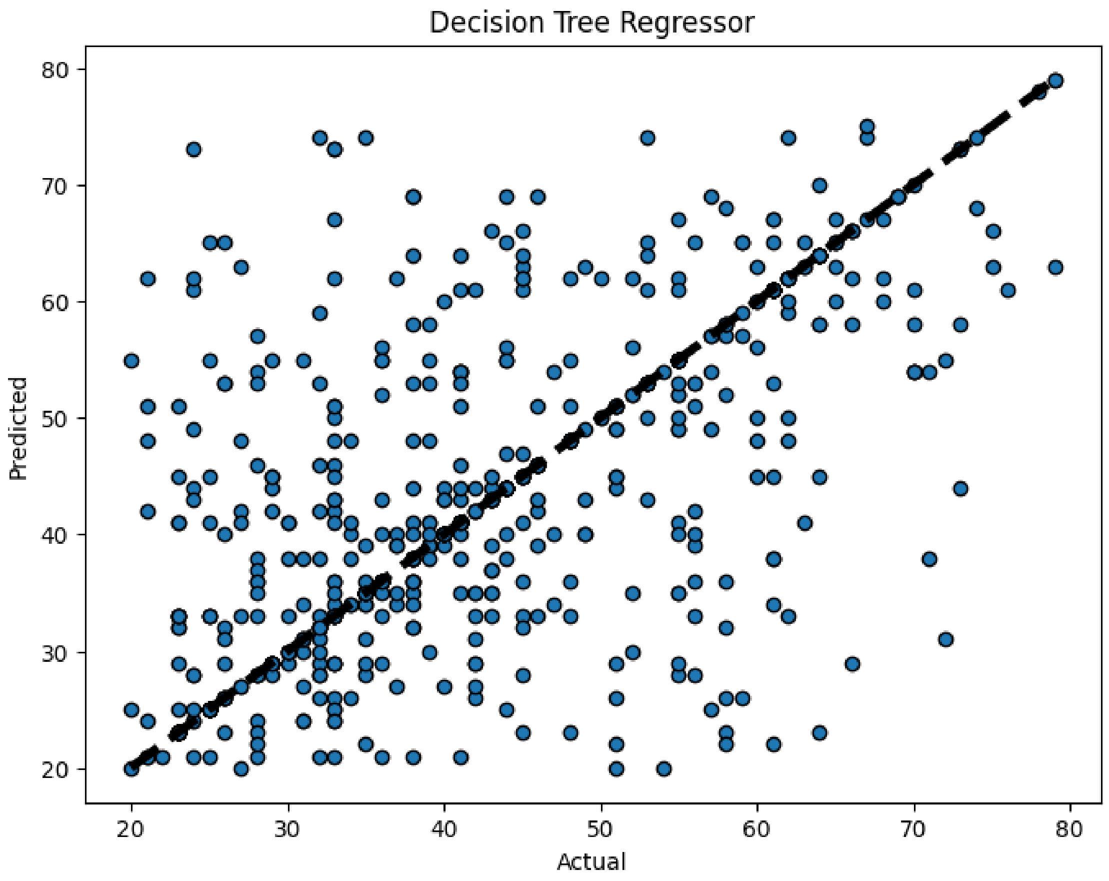
```
Ridge Regression:
MSE: 148.7172173104601
R2: 0.23743376734659571
```



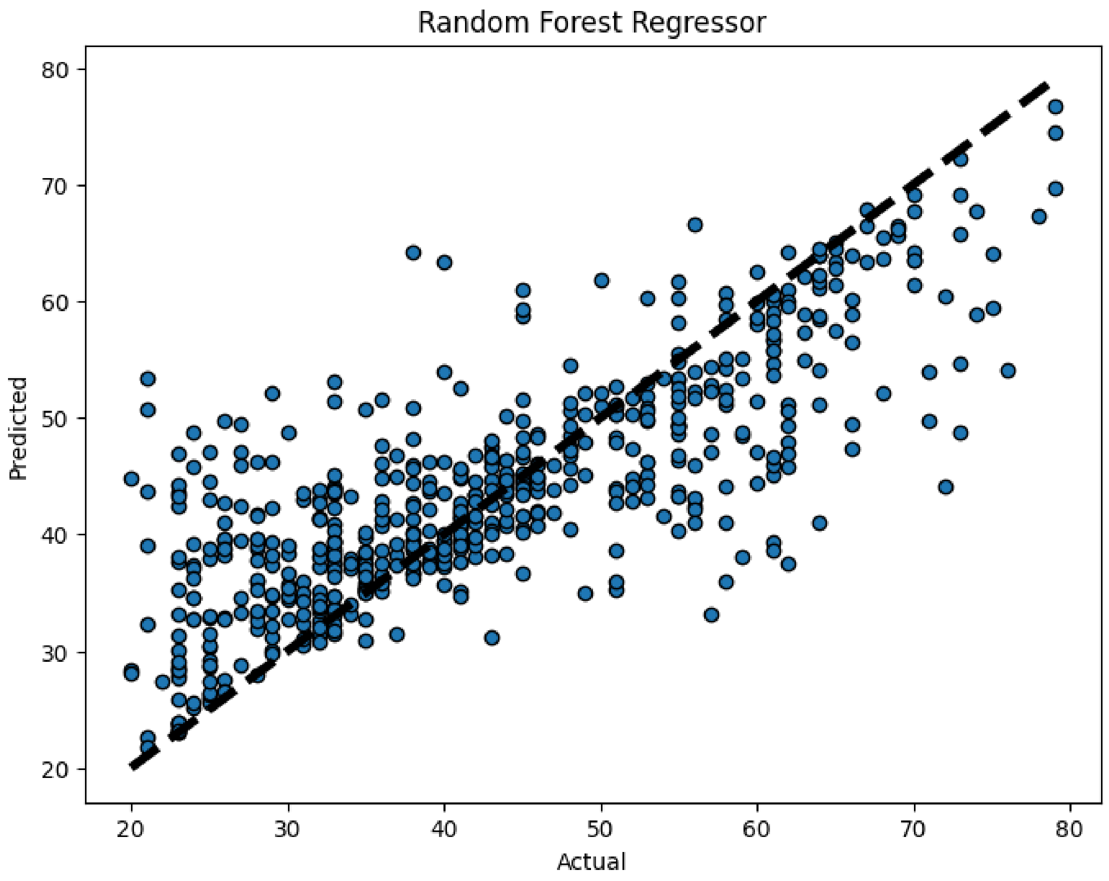
```
Lasso Regression:
MSE: 158.19556643719656
R2: 0.18883234031572071
```



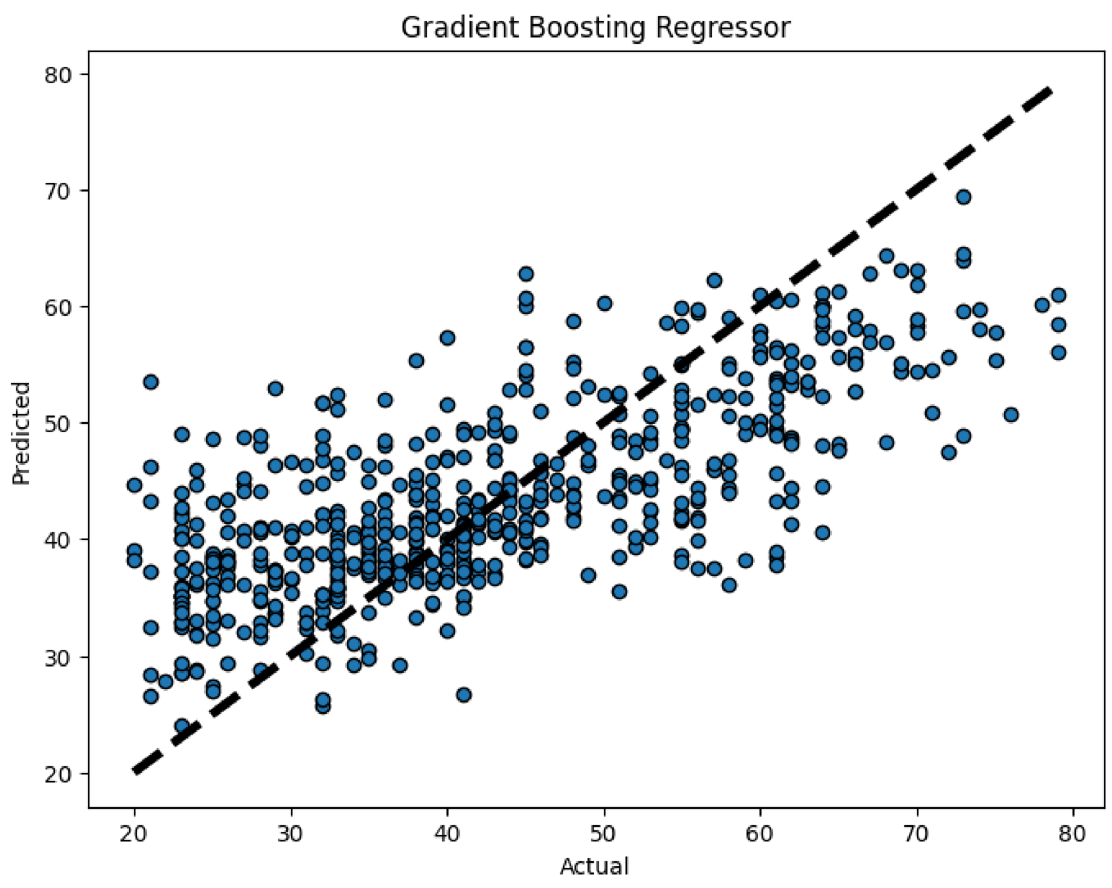
```
Decision Tree Regressor:
MSE: 160.91694352159467
R2: 0.1748781371077185
```



```
Random Forest Regressor:
MSE: 74.1283911960133
R2: 0.6198973526448189
```

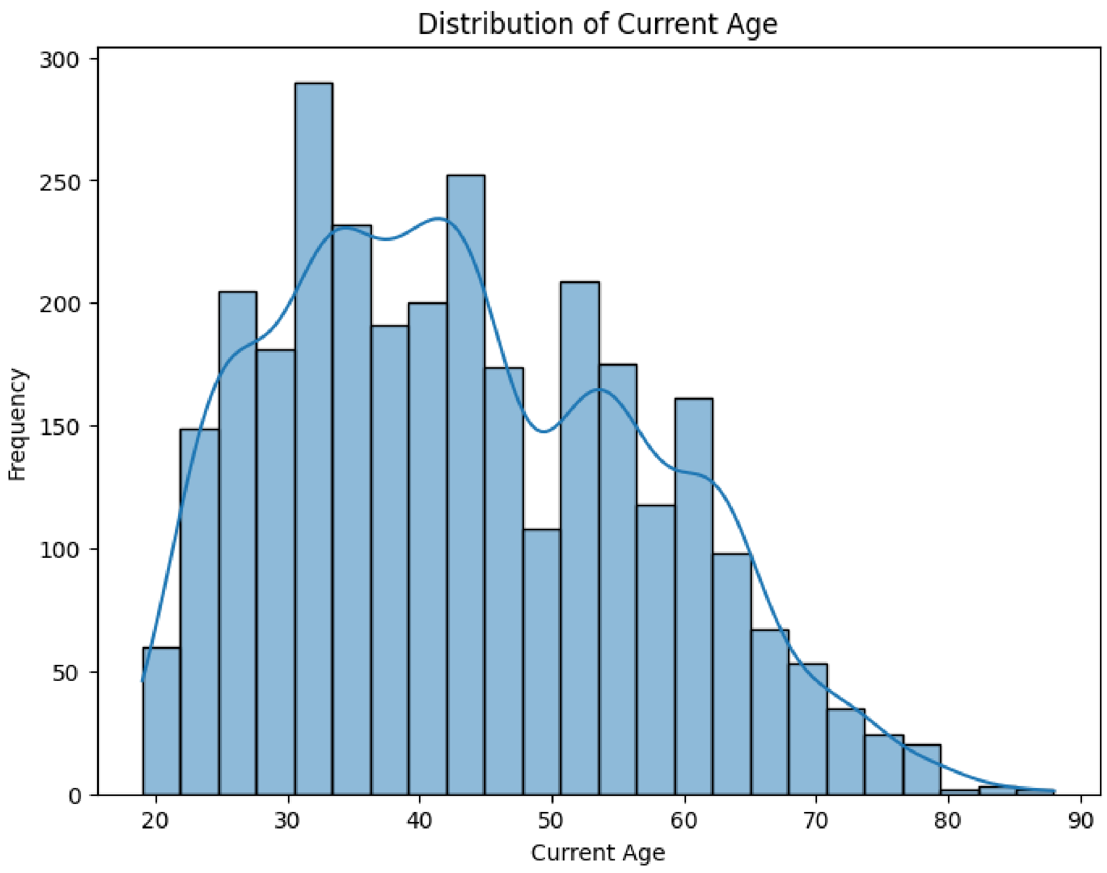


```
Gradient Boosting Regressor:
MSE: 98.92470356193063
R2: 0.49275114290166067
```

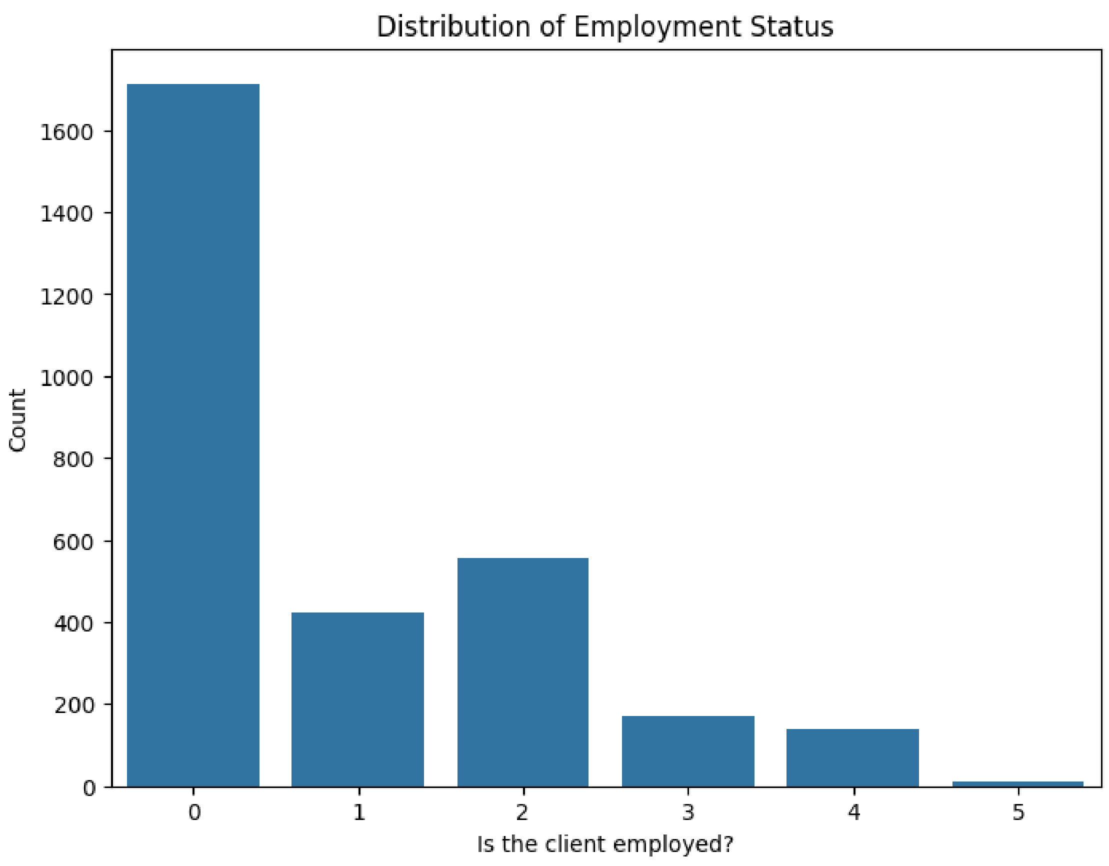


#DATA VISUALIZATION

[9]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot: Distribution of 'Current Age'
plt.figure(figsize=(8, 6))
sns.histplot(df['Current Age'], kde=True)
plt.title('Distribution of Current Age')
plt.xlabel('Current Age')
plt.ylabel('Frequency')
plt.show()
```

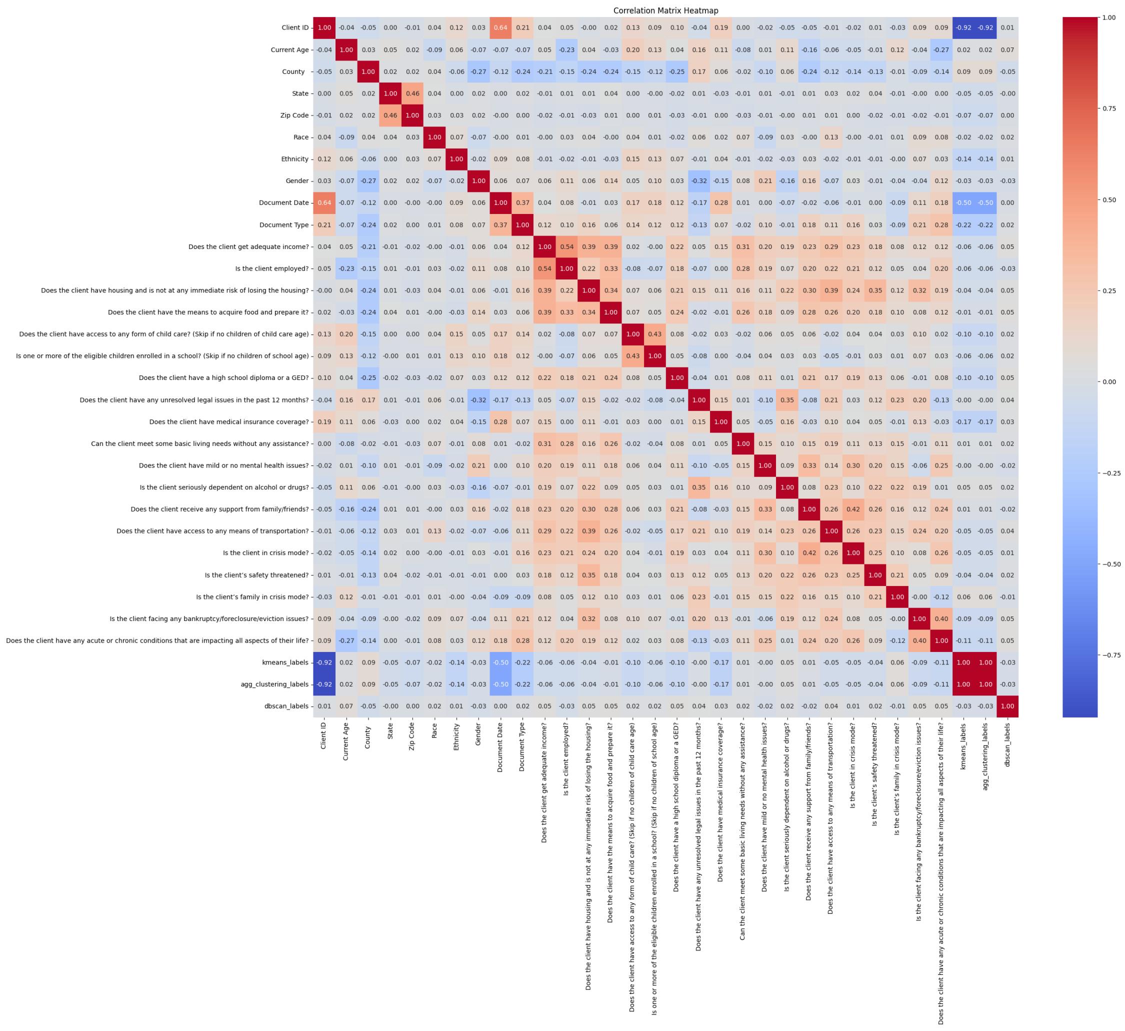


[10]:
```
# Plot: Distribution of 'Is the client employed?'
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Is the client employed?')
plt.title('Distribution of Employment Status')
plt.xlabel('Is the client employed?')
plt.ylabel('Count')
plt.show()
```



- The bar plot shows the distribution of the `Is the client employed?` variable across six categories.
- Most clients fall into category `0.0`, indicating a high number of unemployed clients.
- Categories `1.0`, `2.0`, and `3.0` also have noticeable counts, representing various employment statuses.
- Categories `4.0` and `5.0` have very few clients, indicating these employment statuses are less common.
- This distribution highlights the employment landscape of the clients, with a significant proportion being unemployed.
- Understanding these categories can help in further analysis and targeted interventions.

[11]:
```
# Plot: Heatmap of Correlation Matrix
plt.figure(figsize=(25, 20))
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix Heatmap')
plt.show()
```



0.0.3 Key Points from the Correlation Matrix Heatmap

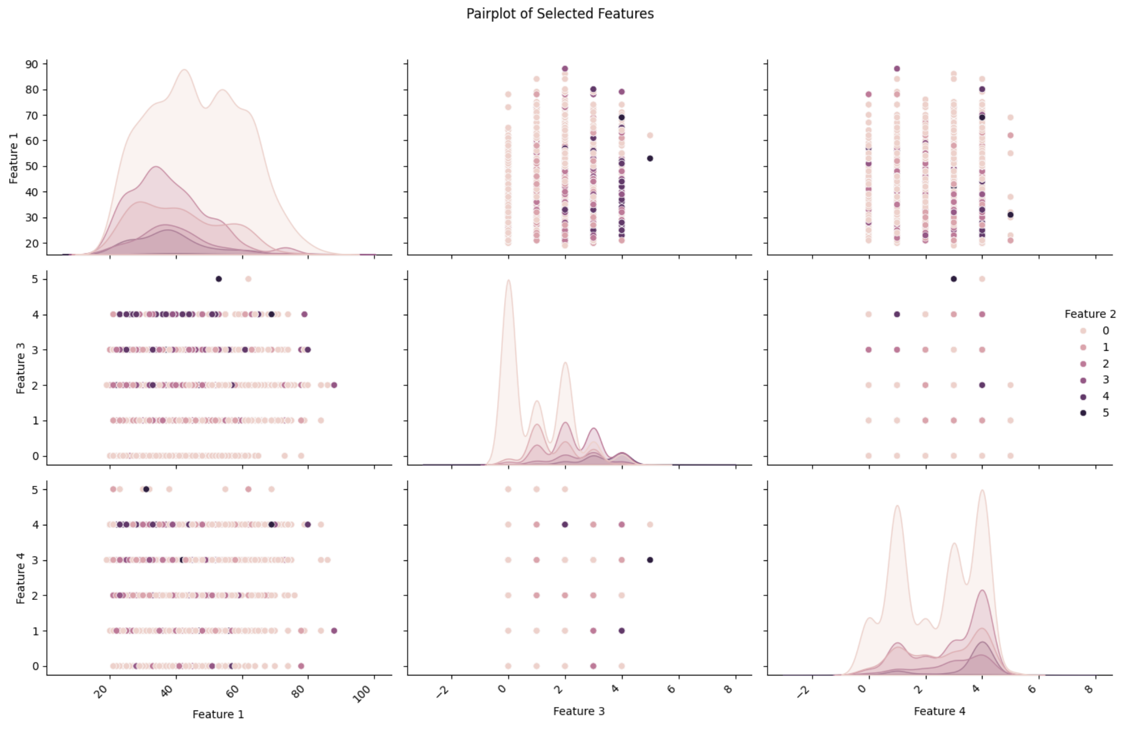
1. **Client ID**:
   - Strong negative correlation with clustering labels (`kmeans_labels`, `agg_clustering_labels`).
2. **Current Age**:
   - Weak correlations with most variables, indicating relative independence.
   - Slight negative correlation with `Is the client employed?` (-0.23).
3. **Adequate Income and Employment**:
   - Moderate positive correlation between `Does the client get adequate income?` and `Is the client employed?` (0.54).
   - Suggests a link between employment status and income adequacy.
4. **Housing Stability**:
   - Moderate positive correlation between `Does the client have housing and is not at any immediate risk of losing the housing?` and income adequacy (0.39).
   - Also shows correlation with employment (0.32).

5. **Document Date**:
    - Negative correlation with clustering labels.
    - Positive correlation with `Client ID` (0.64), possibly due to ordering.
6. **Gender and Clustering Labels**:
    - Moderate negative correlation between `Gender` and clustering labels.
7. **Clustering Labels**:
    - Strong negative correlation with `Client ID` and `Document Date`.
    - Used as derived features in clustering analysis.
8. **Other Notable Correlations**:
    - `Does the client receive any support from family/friends?` has a moderate positive correlation with `Does the client have access to any means of transportation?` (0.42).
    - `Does the client have medical insurance coverage?` correlates with `Can the client meet some basic living needs without any assistance?` (0.31).

These points highlight significant relationships and patterns within the dataset, which can inform further analysis and interventions.

[12]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Define the features and create a mapping from names to numeric labels
selected_features = ['Current Age', 'Is the client employed?',
                     'Does the client get adequate income?',
                     'Does the client have housing and is not at any immediate␣
 ↪risk of losing the housing?']

feature_mapping = {feature: f'Feature {i+1}' for i, feature in␣
 ↪enumerate(selected_features)}

# Replace the column names with the numeric labels in the dataframe
df_numeric = df[selected_features].rename(columns=feature_mapping)

# Create the pairplot with the numeric labels
pairplot = sns.pairplot(df_numeric.dropna(), hue=feature_mapping['Is the client␣
 ↪employed?'], diag_kind='kde', height=3, aspect=1.5)
pairplot.fig.suptitle('Pairplot of Selected Features', y=1.02)

# Adjust the font size of the labels
for ax in pairplot.axes.flatten():
    plt.setp(ax.get_xticklabels(), rotation=45, horizontalalignment='right',␣
 ↪fontsize=10)
    plt.setp(ax.get_yticklabels(), fontsize=10)
    if ax.get_title():
        ax.set_title(ax.get_title(), fontsize=12)

plt.tight_layout()
plt.show()
```

```python
# Display the mapping from numeric labels to original feature names
print("Feature Mapping:")
for original, numeric in feature_mapping.items():
    print(f'{numeric}: {original}')
```



```
Feature Mapping:
Feature 1: Current Age
Feature 2: Is the client employed?
Feature 3: Does the client get adequate income?
Feature 4: Does the client have housing and is not at any immediate risk of
losing the housing?
```

0.0.4 Explanation of the Pairplot Output

The pairplot above provides a visual representation of the relationships between selected features in your dataset. Here's a breakdown of the components and what they indicate:

1. **Diagonal Plots (KDE Plots)**:
   - The diagonal plots show the Kernel Density Estimate (KDE) for each feature, providing a smoothed histogram.
   - **Feature 1 (Current Age)**: Shows the age distribution of the clients. It appears that the majority of clients fall between the ages of 20 and 70.
   - **Feature 2 (Is the client employed?)**: Displays the distribution of employment status. This feature seems to have multiple discrete values (likely representing different

employment states).

- **Feature 3 (Does the client get adequate income?)**: Shows the distribution of income adequacy among clients, indicating several discrete responses.
- **Feature 4 (Does the client have housing and is not at any immediate risk of losing the housing?)**: Illustrates the housing stability of clients, again indicating discrete responses.

2. **Off-Diagonal Plots (Scatter Plots)**:
   - These plots display pairwise relationships between the features using scatter plots.
   - **Feature 1 vs. Feature 2**: Shows how different age groups are distributed across different employment statuses.
   - **Feature 1 vs. Feature 3**: Shows the relationship between age and income adequacy. There appear to be clusters at specific values, indicating the presence of discrete categories in income adequacy.
   - **Feature 1 vs. Feature 4**: Illustrates the relationship between age and housing stability.
   - **Feature 2 vs. Feature 3**: Displays the relationship between employment status and income adequacy, with visible clusters indicating discrete categories.
   - **Feature 2 vs. Feature 4**: Shows the relationship between employment status and housing stability.
   - **Feature 3 vs. Feature 4**: Illustrates the relationship between income adequacy and housing stability, with visible clusters for different categories.
3. **Legend**:
   - The legend indicates the color coding for `Feature 2 (Is the client employed?)`, which has multiple discrete values ranging from 0 to 5. These could represent different employment statuses (e.g., unemployed, part-time, full-time).
4. **Feature Mapping**:
   - Below the plot, the mapping from numeric labels to the original feature names is provided for reference:
     - Feature 1: Current Age
     - Feature 2: Is the client employed?
     - Feature 3: Does the client get adequate income?
     - Feature 4: Does the client have housing and is not at any immediate risk of losing the housing?

0.0.5 Key Observations

- **Age Distribution**: Most clients are aged between 20 and 70, with peaks indicating common age ranges.
- **Discrete Features**: Features 2, 3, and 4 have discrete values, indicating categorical responses.
- **Pairwise Relationships**: The scatter plots show clusters indicating relationships between different features. For example, older clients might have different employment statuses compared to younger clients.
- **Multicollinearity Check**: The heatmap of the correlation matrix can further help identify if there are strong linear relationships between features.

This plot helps you to understand the distributions and relationships between different features in your dataset, providing insights into potential patterns and correlations.

```
[ ]:
```