Homework 4 - Final

Matthew Henao Z23685608

Part 1: Problem Understanding

Part 2: Data Preparation

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from statsmodels.stats.outliers_influence import
variance_inflation_factor
import statsmodels.api as sm
from statsmodels.discrete.discrete_model import Logit
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from statsmodels.tools.tools import add_constant

from sklearn import datasets
from sklearn.linear_model import LinearRegression, Lasso, LassoCV
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import cross_val_score, train_test_split,
KFold
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import silhouette_score, davies_bouldin_score,
mean_squared_error, r2_score
from sklearn.tree import DecisionTreeClassifier,
DecisionTreeRegressor, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
from sklearn.pipeline import Pipeline

data = pd.read_csv('BostonHousing.csv')

data.head()
```

```
      crim    zn  indus  chas    nox     rm   age     dis  rad  tax
ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
```

```
17.8
2  0.02729   0.0   7.07      0  0.469  7.185  61.1  4.9671    2  242
17.8
3  0.03237   0.0   2.18      0  0.458  6.998  45.8  6.0622    3  222
18.7
4  0.06905   0.0   2.18      0  0.458  7.147  54.2  6.0622    3  222
18.7

        b  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    int64
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

```
data = data.dropna()
```

Part 3: Exploratory Data Analysis

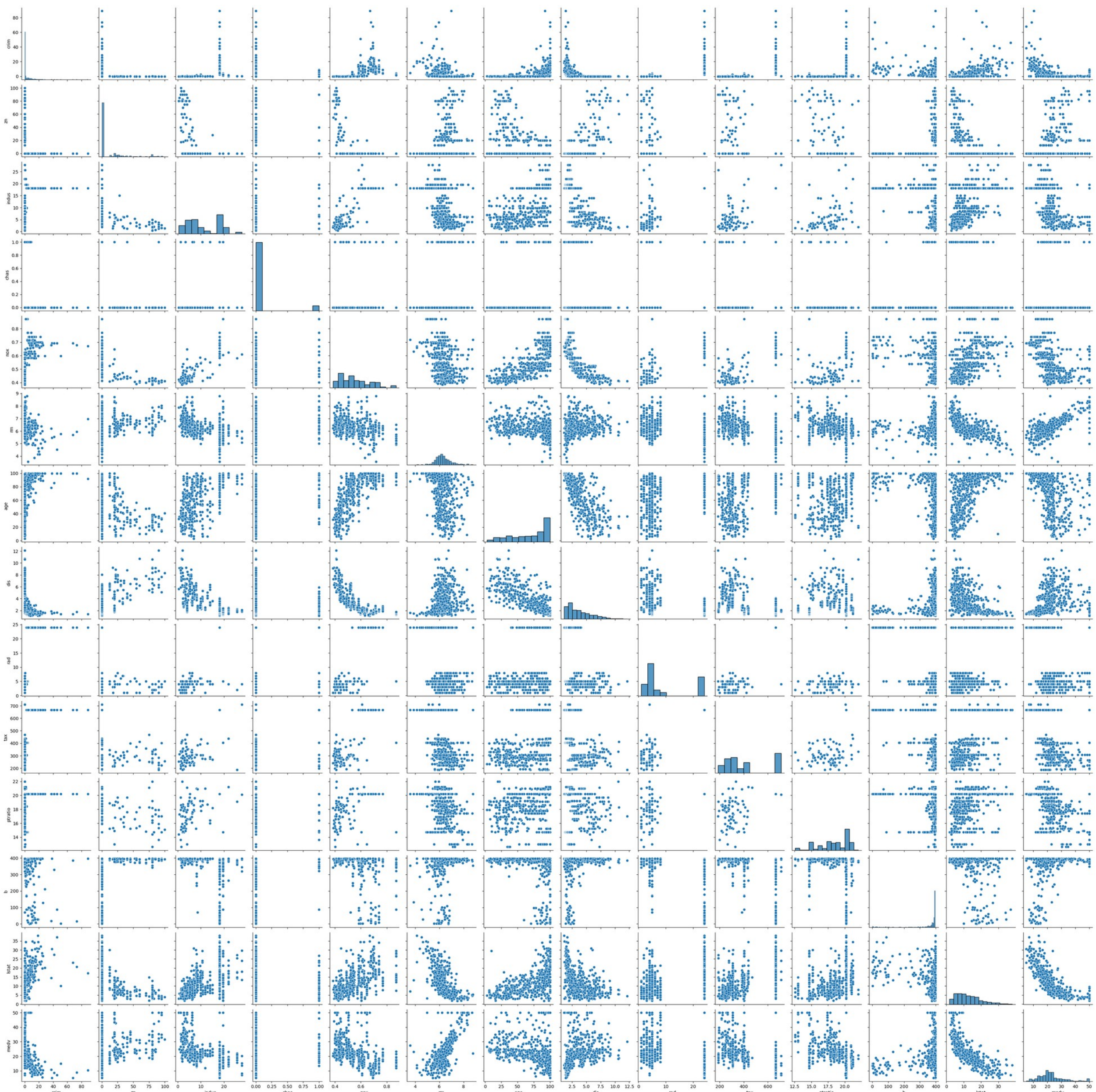
```
data.describe()
```

```
             crim          zn       indus        chas         nox
rm  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695
```

```
6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878
0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000
3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000
5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000
6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000
6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000
8.780000

              age         dis         rad         tax     ptratio
b  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534
356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946
91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000
0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000
375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000
391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000
396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000
396.900000

            lstat        medv
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

```
sns.pairplot(data)
plt.show()
```

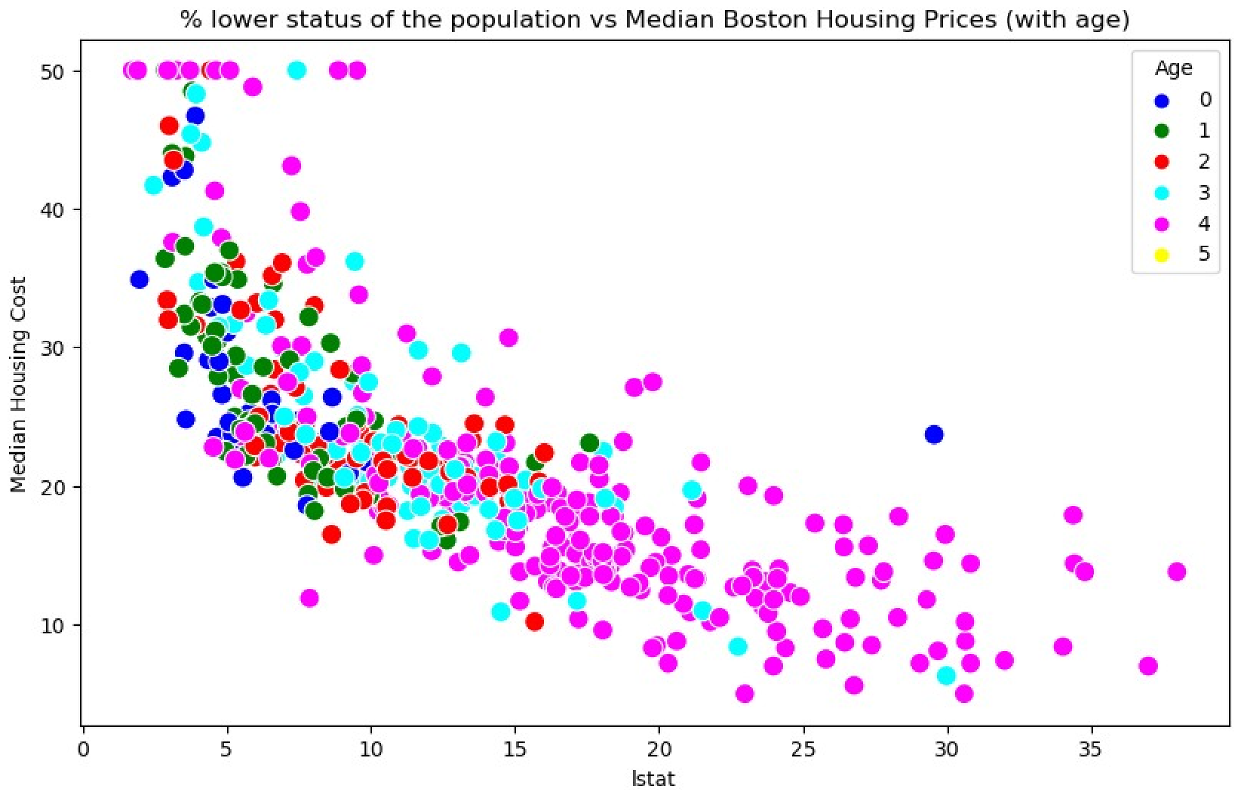
```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has
changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



```
X = add_constant(data.drop(columns=['age']))

vif_data = pd.DataFrame()
vif_data["Variable"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]

print(vif_data)

   Variable         VIF
0     const  641.736995
1      crim    1.831512
```

```
2         zn    2.318518
3      indus    3.992499
4       chas    1.092490
5        nox    4.276892
6         rm    2.178473
7        dis    4.062124
8        rad    7.764635
9        tax    9.197272
10  ptratio    1.981797
11         b    1.376049
12     lstat    3.237109
13      medv    3.855663
```

```
bins = [0, 20, 40, 60, 80, 100, 120]
labels = [0, 1, 2, 3, 4, 5]
data['age_group'] = pd.cut(data['age'], bins=bins, labels=labels,
include_lowest=True)

palette = {
    0: 'blue',
    1: 'green',
    2: 'red',
    3: 'cyan',
    4: 'magenta',
    5: 'yellow'
}

plt.figure(figsize=(10, 6))
sns.scatterplot(x='lstat', y='medv', hue='age_group', data=data,
palette=palette, s=100)
plt.title('% lower status of the population vs Median Boston Housing
Prices (with age)')
plt.xlabel('lstat')
plt.ylabel('Median Housing Cost')
plt.legend(title='Age')
plt.show()
```



Step 4: Setup Phase

```
#Already completed :)
```

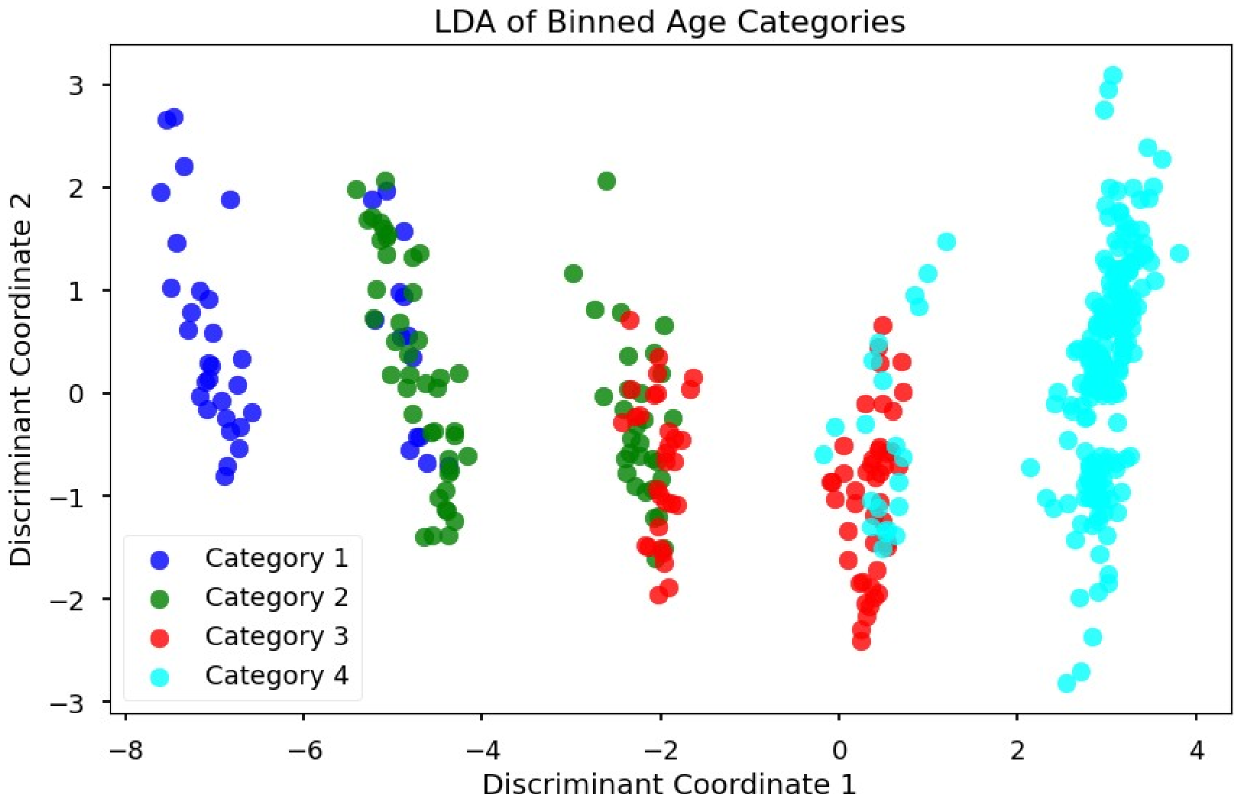
Step 5: Modeling Phase

```
X = data[['crim', 'zn', 'indus', 'chas', 'nox',
               'rm', 'age', 'dis', 'rad', 'tax',
               'ptratio', 'lstat']]
X = sm.add_constant(X)
y = data['lstat']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

linear_reg = LinearRegression()
linear_reg.fit(X_scaled, y_train)

X_test_scaled = scaler.transform(X_test)

y_pred = linear_reg.predict(X_test_scaled)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f"Coefficients: {linear_reg.coef_}")
print(f"Intercept: {linear_reg.intercept_}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R2): {r2}")
```

```
Coefficients: [ 0.00000000e+00  0.00000000e+00 -1.24028298e-14
1.29692320e-14
 -1.09171619e-16 -5.19221872e-15  3.20937440e-15  5.32670209e-15
  1.70378279e-14  2.51833160e-15 -3.22635759e-15 -3.96874851e-15
  7.10157559e+00]
Intercept: 12.457351485148518
Mean Squared Error (MSE): 2.151096078685384e-28
R-squared (R2): 1.0
```

```
bins = np.linspace(min(data['age']), max(data['age']), num=5)
age_categories = pd.cut(data['age'], bins, labels=False,
include_lowest=True)
data['age_binned'] = age_categories

X = data.drop(['age', 'age_binned'], axis=1)
y = data['age_binned']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)

X_r_lda = lda.transform(X_train)

target_names = ['Category 1', 'Category 2', 'Category 3', 'Category
4']
colors = ['blue', 'green', 'red', 'cyan']

with plt.style.context('seaborn-talk'):
    fig, ax = plt.subplots(figsize=[10,6])
    for color, i, target_name in zip(colors, range(len(target_names)),
target_names):
        ax.scatter(X_r_lda[y_train == i, 0], X_r_lda[y_train == i, 1],
alpha=0.8, color=color, label=target_name)
    ax.set_title('LDA of Binned Age Categories')
    ax.set_xlabel('Discriminant Coordinate 1')
    ax.set_ylabel('Discriminant Coordinate 2')
    ax.legend(loc='best')
    plt.show()
```




```
y_pred = lda.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))

Accuracy: 0.8627450980392157
              precision    recall  f1-score   support

           0       1.00      0.67      0.80         9
           1       0.80      0.70      0.74        23
           2       0.68      0.85      0.76        20
           3       0.96      0.98      0.97        50

    accuracy                           0.86       102
   macro avg       0.86      0.80      0.82       102
weighted avg       0.87      0.86      0.86       102


poly3_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly', PolynomialFeatures(degree=3)),
    ('linear', LinearRegression())
])
```

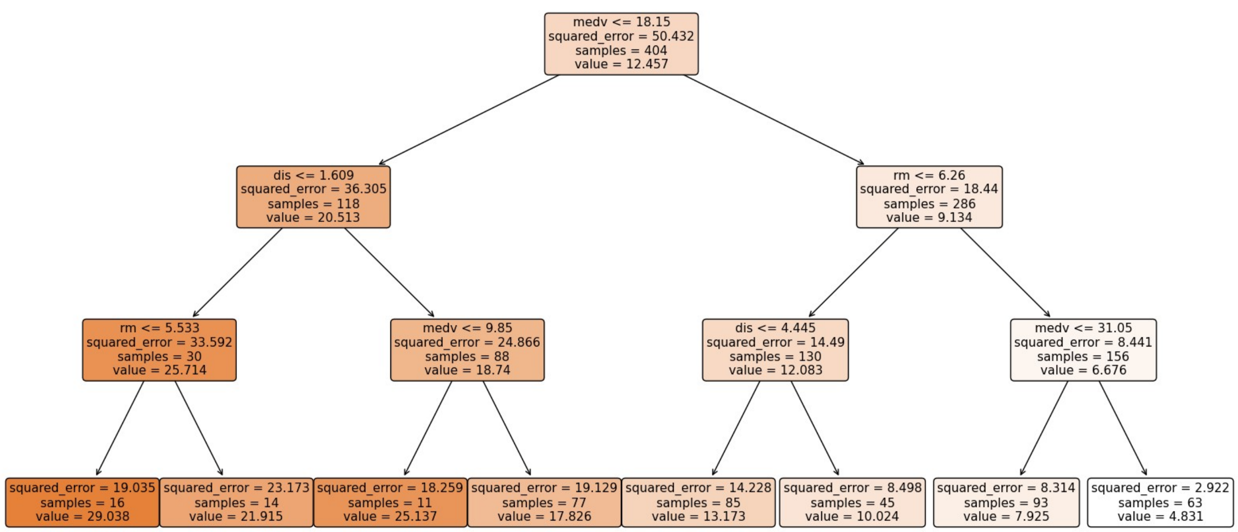
```python
poly3_pipeline.fit(X_train, y_train)

y_pred_poly3 = poly3_pipeline.predict(X_test)

mse_poly3 = mean_squared_error(y_test, y_pred_poly3)

print(f"Mean Squared Error for 3rd-degree Polynomial:
{mse_poly3:.2f}")

poly4_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly', PolynomialFeatures(degree=4)),
    ('linear', LinearRegression())
])

poly4_pipeline.fit(X_train, y_train)

y_pred_poly4 = poly4_pipeline.predict(X_test)

mse_poly4 = mean_squared_error(y_test, y_pred_poly4)

print(f"Mean Squared Error for 4th-degree Polynomial:
{mse_poly4:.2f}")

poly5_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly', PolynomialFeatures(degree=5)),
    ('linear', LinearRegression())
])

poly5_pipeline.fit(X_train, y_train)

y_pred_poly5 = poly5_pipeline.predict(X_test)

mse_poly5 = mean_squared_error(y_test, y_pred_poly5)

print(f"Mean Squared Error for 5th-degree Polynomial:
{mse_poly5:.2f}")
```

```
Mean Squared Error for 3rd-degree Polynomial: 36.52
Mean Squared Error for 4th-degree Polynomial: 19.68
Mean Squared Error for 5th-degree Polynomial: 35.87
```

```python
X = data.drop('lstat', axis=1)
y = data['lstat']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

dtc = DecisionTreeRegressor(max_depth=3, random_state=42)
```

```python
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)

feature_names = X.columns.tolist()
plt.figure(figsize=(20,10))
plot_tree(dtc, filled=True, feature_names=feature_names, rounded=True)
plt.show()
```

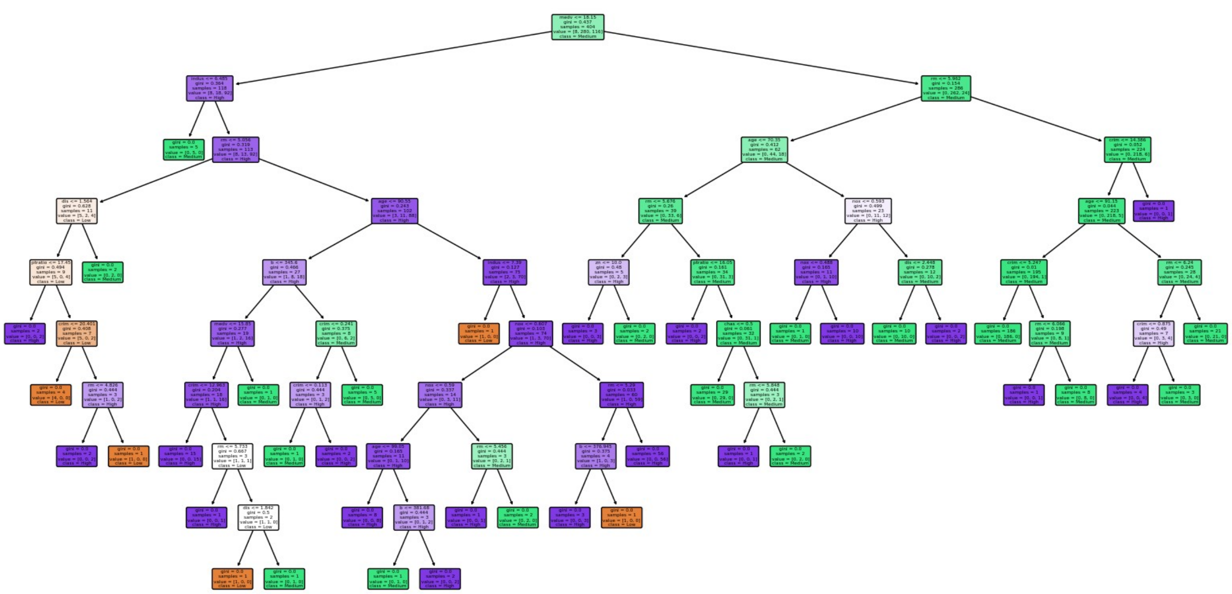
```
Mean Squared Error: 13.993585825348177
R-squared: 0.7307356252630821
```




```python
data['MEDV_category'] = pd.cut(data['lstat'], bins=[0, 15, 30, 50],
labels=['Low', 'Medium', 'High'])

X = data.drop(['lstat', 'MEDV_category'], axis=1)
y = data['MEDV_category']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

dtc = DecisionTreeClassifier(max_depth=10, random_state=42)

dtc.fit(X_train, y_train)
```

```
y_pred = dtc.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))

feature_names = data.columns.drop(['lstat', 'MEDV_category']).tolist()
# Actual feature names from the dataset
class_names = ['Low', 'Medium', 'High']
plt.figure(figsize=(20,10))
plot_tree(dtc, filled=True, feature_names=feature_names,
class_names=class_names, rounded=True)
plt.show()
```

```
Accuracy: 0.7941176470588235
              precision    recall  f1-score   support

        High       1.00      0.25      0.40         4
         Low       0.81      0.89      0.85        64
      Medium       0.74      0.68      0.71        34

    accuracy                           0.79       102
   macro avg       0.85      0.61      0.65       102
weighted avg       0.80      0.79      0.79       102
```



```
data = pd.get_dummies(data)

X = data.drop('lstat', axis=1)
```

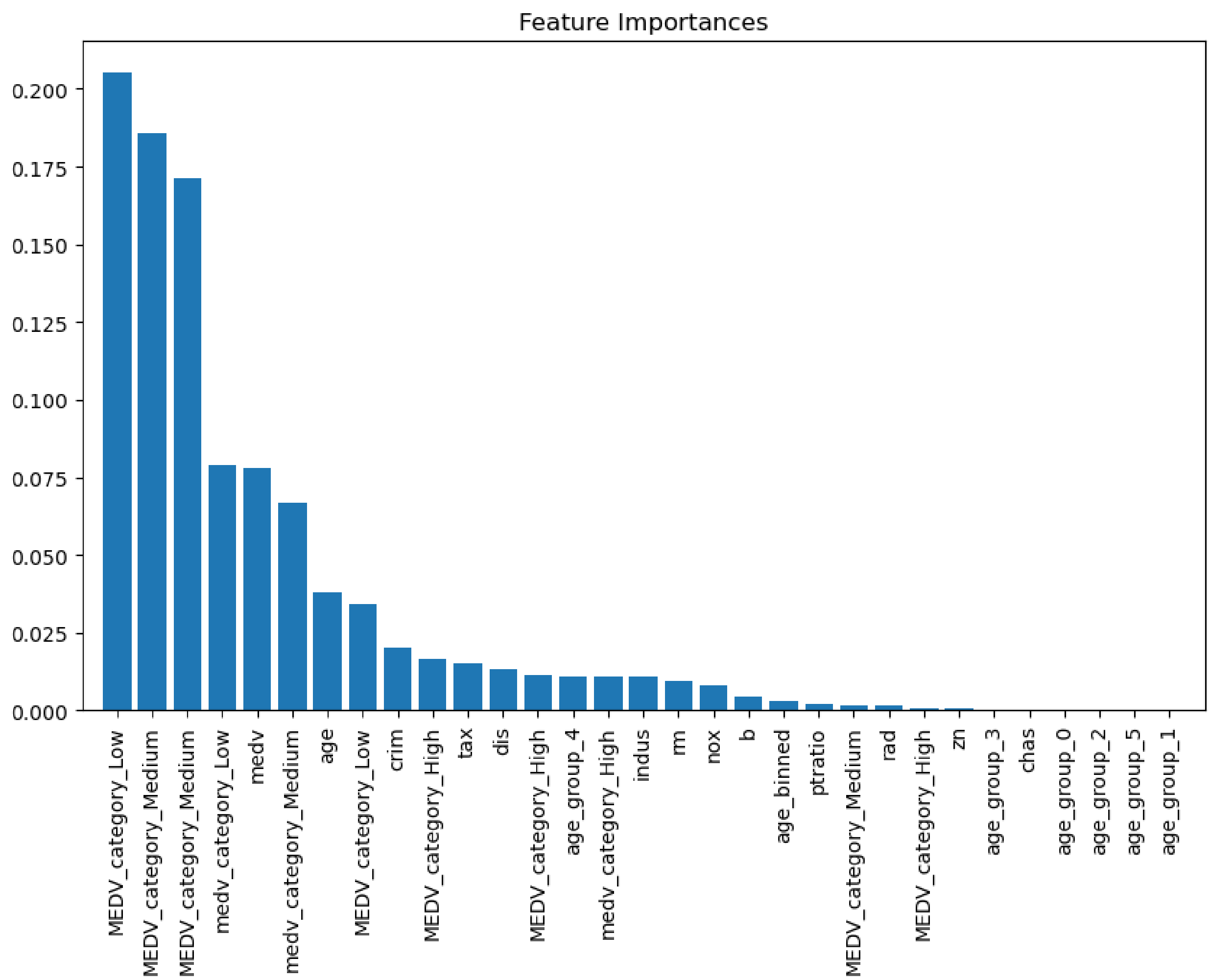
```python
y = data['lstat']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
labels = kmeans.labels_

feature_names = X_train.columns.tolist()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/cluster/_kmeans.py:1412: 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)
```

```python
data['MEDV_category'] = pd.cut(data['lstat'], bins=[0, 15, 30, 50],
labels=['Low', 'Medium', 'High'])

X = data.drop(['lstat', 'MEDV_category'], axis=1)
y = data['MEDV_category']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

clf = RandomForestClassifier(n_estimators=100, random_state=42)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print(classification_report(y_test, y_pred))

feature_importances = clf.feature_importances_
features = X.columns.tolist()
indices = np.argsort(feature_importances)[::-1]

plt.figure(figsize=(10,6))
plt.title("Feature Importances")
plt.bar(range(X_train.shape[1]), feature_importances[indices],
align="center")
plt.xticks(range(X_train.shape[1]), [features[i] for i in indices],
rotation=90)
```

```
plt.xlim([-1, X_train.shape[1]])
plt.show()
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

        High       1.00      1.00      1.00         4
         Low       1.00      1.00      1.00        64
      Medium       1.00      1.00      1.00        34

    accuracy                           1.00       102
   macro avg       1.00      1.00      1.00       102
weighted avg       1.00      1.00      1.00       102
```




```
print(data.info())

data = pd.get_dummies(data, drop_first=True)
```

```
data = data.astype(float)

X = data.drop('lstat', axis=1)
y = data['lstat']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 33 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   crim                  506 non-null    float64
 1   zn                    506 non-null    float64
 2   indus                 506 non-null    float64
 3   chas                  506 non-null    float64
 4   nox                   506 non-null    float64
 5   rm                    506 non-null    float64
 6   age                   506 non-null    float64
 7   dis                   506 non-null    float64
 8   rad                   506 non-null    float64
 9   tax                   506 non-null    float64
 10  ptratio               506 non-null    float64
 11  b                     506 non-null    float64
 12  lstat                 506 non-null    float64
 13  medv                  506 non-null    float64
 14  age_binned            506 non-null    float64
 15  age_group_0           506 non-null    float64
 16  age_group_1           506 non-null    float64
 17  age_group_2           506 non-null    float64
 18  age_group_3           506 non-null    float64
 19  age_group_4           506 non-null    float64
 20  age_group_5           506 non-null    float64
 21  MEDV_category_Low     506 non-null    float64
 22  MEDV_category_Medium  506 non-null    float64
 23  MEDV_category_High    506 non-null    float64
 24  MEDV_category_Medium  506 non-null    float64
 25  MEDV_category_High    506 non-null    float64
 26  medv_category_Low     506 non-null    bool
 27  medv_category_Medium  506 non-null    bool
 28  medv_category_High    506 non-null    bool
 29  MEDV_category_Low     506 non-null    bool
```

```
 30  MEDV_category_Medium  506 non-null    bool
 31  MEDV_category_High    506 non-null    bool
 32  MEDV_category         506 non-null    category
dtypes: bool(6), category(1), float64(26)
memory usage: 106.5 KB
None
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412: 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)
```

```
KMeans(n_clusters=3, random_state=42)
```

```
X = data.drop('lstat', axis=1)
y = data['lstat']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
labels = kmeans.labels_

feature_names = X_train.columns.tolist()
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412: 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)
```

```
data = pd.get_dummies(data)

X = data.drop('lstat', axis=1)
y = data['lstat']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
labels = kmeans.labels_
```

```
def cluster_profiling(X, labels, feature_names):
    df = pd.DataFrame(X, columns=feature_names)
    df['Cluster'] = labels
    profile = df.groupby('Cluster').mean()
    return profile

def feature_importance(kmeans, feature_names):
    centroids = kmeans.cluster_centers_
    importance = pd.DataFrame(centroids, columns=feature_names).abs()
    return importance

def cluster_validation(X, labels):
    silhouette_avg = silhouette_score(X, labels)
    davies_bouldin = davies_bouldin_score(X, labels)
    return silhouette_avg, davies_bouldin

def anomaly_detection(X, kmeans):
    distances = cdist(X, kmeans.cluster_centers_, 'euclidean')
    distance_to_nearest_centroid = np.min(distances, axis=1)
    outlier_threshold = np.percentile(distance_to_nearest_centroid,
95)
    anomaly_indices = np.where(distance_to_nearest_centroid >
outlier_threshold)[0]
    return anomaly_indices

feature_names = X.columns.tolist()

profile = cluster_profiling(X_scaled, labels, feature_names)
print("Cluster Profiling: (Predictor mean by clusters)\n", profile)

importance = feature_importance(kmeans, feature_names)
print("\nFeature Importance: (Just absolute values)\n", importance)

silhouette_avg, davies_bouldin = cluster_validation(X_scaled, labels)
print("\nCluster Validation Metrics:")
print("Silhouette Score:", silhouette_avg)
print("Davies-Bouldin Score:", davies_bouldin)

anomaly_indices = anomaly_detection(X_scaled, kmeans)
print("\nAnomaly Detection: (Distance threshold to the centroid for
95% of the data points)")
print("Indices of Anomalies in the Original Dataset:",
anomaly_indices)

anomalies = pd.DataFrame(X,
columns=feature_names).iloc[anomaly_indices]
print("\nData of Detected Anomalies:\n", anomalies)
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/cluster/_kmeans.py:1412: 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)

Cluster Profiling: (Predictor mean by clusters)
              crim        zn     indus      chas       nox        rm
age  \
Cluster

0         0.603081 -0.483294  0.837315 -0.034053  0.850137 -0.514199
0.773717
1        -0.316510  0.243052 -0.422046  0.024347 -0.426541  0.297773 -
0.393679
2         1.068749 -0.500320  0.898908 -0.278089  0.846520 -1.849691
0.954697

              dis       rad       tax  ...  MEDV_category_Medium  \
Cluster                                ...
0        -0.676787  0.722075  0.802926  ...              1.384232
1         0.345093 -0.373902 -0.411191  ...             -0.634648
2        -0.859746  1.109508  1.079841  ...             -0.634648

         MEDV_category_High  medv_category_Low
medv_category_Medium  \
Cluster

0                 -0.142134          -1.275569              1.178315

1                 -0.142134           0.641126             -0.536514

2                  7.035624          -1.308208             -0.665475


         medv_category_High  MEDV_category_Low
MEDV_category_Medium  \
Cluster

0                  0.298735           0.857920             -0.323445

1                 -0.256833          -0.466327              0.191404

2                  3.893584           2.060765             -1.301281


         MEDV_category_High  MEDV_category_Medium  MEDV_category_High

Cluster

0                 -0.469621              1.384232           -0.142134

1                  0.235683             -0.634648           -0.142134
```

```
2                 -0.469621            -0.634648           7.035624


[3 rows x 33 columns]

Feature Importance: (Just absolute values)
        crim        zn     indus      chas       nox        rm
age  \
0  0.603081  0.483294  0.837315  0.034053  0.850137  0.514199
0.773717
1  0.316510  0.243052  0.422046  0.024347  0.426541  0.297773
0.393679
2  1.068749  0.500320  0.898908  0.278089  0.846520  1.849691
0.954697

        dis       rad       tax  ...  MEDV_category_Medium  \
0  0.676787  0.722075  0.802926  ...              1.384232
1  0.345093  0.373902  0.411191  ...              0.634648
2  0.859746  1.109508  1.079841  ...              0.634648

   MEDV_category_High  medv_category_Low  medv_category_Medium  \
0            0.142134           1.275569              1.178315
1            0.142134           0.641126              0.536514
2            7.035624           1.308208              0.665475

   medv_category_High  MEDV_category_Low  MEDV_category_Medium  \
0            0.298735           0.857920              0.323445
1            0.256833           0.466327              0.191404
2            3.893584           2.060765              1.301281

   MEDV_category_High  MEDV_category_Medium  MEDV_category_High
0            0.469621              1.384232            0.142134
1            0.235683              0.634648            0.142134
2            0.469621              0.634648            7.035624

[3 rows x 33 columns]

Cluster Validation Metrics:
Silhouette Score: 0.32842691255071504
Davies-Bouldin Score: 1.0243279485067176

Anomaly Detection: (Distance threshold to the centroid for 95% of the
data points)
Indices of Anomalies in the Original Dataset: [ 20  57  86  99 105 106
111 126 135 171 176 188 193 222 226 236 270 277
 332 388 392]

Data of Detected Anomalies:
          crim     zn  indus  chas     nox     rm    age     dis   rad
```

```
tax  \
20    1.25179    0.0   8.14   0.0  0.5380  5.570   98.1  3.7979   4.0
307.0
57    0.01432  100.0   1.32   0.0  0.4110  6.816   40.5  8.3248   5.0
256.0
86    0.05188    0.0   4.49   0.0  0.4490  6.015   45.1  4.4272   3.0
247.0
99    0.06860    0.0   2.89   0.0  0.4450  7.416   62.5  3.4952   2.0
276.0
105   0.13262    0.0   8.56   0.0  0.5200  5.851   96.7  2.1069   5.0
384.0
106   0.17120    0.0   8.56   0.0  0.5200  5.836   91.9  2.2110   5.0
384.0
111   0.10084    0.0  10.01   0.0  0.5470  6.715   81.6  2.6775   6.0
432.0
126   0.38735    0.0  25.65   0.0  0.5810  5.613   95.6  1.7572   2.0
188.0
135   0.55778    0.0  21.89   0.0  0.6240  6.335   98.2  2.1107   4.0
437.0
171   2.31390    0.0  19.58   0.0  0.6050  5.880   97.3  2.3887   5.0
403.0
176   0.07022    0.0   4.05   0.0  0.5100  6.020   47.2  3.5549   5.0
296.0
188   0.12579   45.0   3.44   0.0  0.4370  6.556   29.1  4.5667   5.0
398.0
193   0.02187   60.0   2.93   0.0  0.4010  6.800    9.9  6.2196   1.0
265.0
222   0.62356    0.0   6.20   1.0  0.5070  6.879   77.7  3.2721   8.0
307.0
226   0.38214    0.0   6.20   0.0  0.5040  8.040   86.5  3.2157   8.0
307.0
236   0.52058    0.0   6.20   1.0  0.5070  6.631   76.5  4.1480   8.0
307.0
270   0.29916   20.0   6.96   0.0  0.4640  5.856   42.1  4.4290   3.0
223.0
277   0.06127   40.0   6.41   1.0  0.4470  6.826   27.6  4.8628   4.0
254.0
332   0.03466   35.0   6.06   0.0  0.4379  6.031   23.3  6.6407   1.0
304.0
388  14.33370    0.0  18.10   0.0  0.7000  4.880  100.0  1.5895  24.0
666.0
392  11.57790    0.0  18.10   0.0  0.7000  5.036   97.0  1.7700  24.0
666.0

     ...  MEDV_category_Medium  MEDV_category_High  medv_category_Low
\
20   ...                   1.0                 0.0                0.0

57   ...                   0.0                 0.0                1.0
```

```
86   ...                     0.0                   0.0                  1.0

99   ...                     0.0                   0.0                  1.0

105  ...                     1.0                   0.0                  0.0

106  ...                     1.0                   0.0                  0.0

111  ...                     0.0                   0.0                  1.0

126  ...                     1.0                   0.0                  0.0

135  ...                     1.0                   0.0                  0.0

171  ...                     0.0                   0.0                  1.0

176  ...                     0.0                   0.0                  1.0

188  ...                     0.0                   0.0                  1.0

193  ...                     0.0                   0.0                  1.0

222  ...                     0.0                   0.0                  1.0

226  ...                     0.0                   0.0                  1.0

236  ...                     0.0                   0.0                  1.0

270  ...                     0.0                   0.0                  1.0

277  ...                     0.0                   0.0                  1.0

332  ...                     0.0                   0.0                  1.0

388  ...                     0.0                   1.0                  0.0

392  ...                     1.0                   0.0                  0.0


     medv_category_Medium  medv_category_High  MEDV_category_Low  \
20                    1.0                 0.0                1.0
57                    0.0                 0.0                0.0
86                    0.0                 0.0                0.0
99                    0.0                 0.0                0.0
105                   1.0                 0.0                0.0
106                   1.0                 0.0                0.0
111                   0.0                 0.0                0.0
126                   0.0                 1.0                0.0
135                   1.0                 0.0                0.0
171                   0.0                 0.0                0.0
176                   0.0                 0.0                0.0
```

```
188                    0.0                  0.0                  0.0
193                    0.0                  0.0                  0.0
222                    0.0                  0.0                  0.0
226                    0.0                  0.0                  0.0
236                    0.0                  0.0                  0.0
270                    0.0                  0.0                  0.0
277                    0.0                  0.0                  0.0
332                    0.0                  0.0                  0.0
388                    0.0                  1.0                  1.0
392                    1.0                  0.0                  1.0

     MEDV_category_Medium  MEDV_category_High  MEDV_category_Medium  \
20                    0.0                 0.0                   1.0
57                    0.0                 1.0                   0.0
86                    1.0                 0.0                   0.0
99                    0.0                 1.0                   0.0
105                   1.0                 0.0                   1.0
106                   1.0                 0.0                   1.0
111                   1.0                 0.0                   0.0
126                   1.0                 0.0                   1.0
135                   1.0                 0.0                   1.0
171                   1.0                 0.0                   0.0
176                   1.0                 0.0                   0.0
188                   1.0                 0.0                   0.0
193                   0.0                 1.0                   0.0
222                   1.0                 0.0                   0.0
226                   0.0                 1.0                   0.0
236                   1.0                 0.0                   0.0
270                   1.0                 0.0                   0.0
277                   0.0                 1.0                   0.0
332                   1.0                 0.0                   0.0
388                   0.0                 0.0                   0.0
392                   0.0                 0.0                   1.0

     MEDV_category_High
20                  0.0
57                  0.0
86                  0.0
99                  0.0
105                 0.0
106                 0.0
111                 0.0
126                 0.0
135                 0.0
171                 0.0
176                 0.0
188                 0.0
193                 0.0
222                 0.0
```

```
226                 0.0
236                 0.0
270                 0.0
277                 0.0
332                 0.0
388                 1.0
392                 0.0

[21 rows x 33 columns]
```

Step 6: Evaluation Phase

```
X = data.drop('lstat', axis=1)
y = data['lstat']

lr = LinearRegression()

cv_strategy = KFold(n_splits=5, shuffle=True, random_state=42)

cv_scores = cross_val_score(lr, X, y, cv=cv_strategy,
scoring='neg_mean_squared_error')

cv_scores = -cv_scores

print(f"Cross-validation scores for each fold: {cv_scores}")
print(f"Mean cross-validation score: {cv_scores.mean()}")
print(f"Standard deviation of cross-validation scores:
{cv_scores.std()}")

Cross-validation scores for each fold: [4.5798948  4.56838833
5.10731359 4.406928   4.99477415]
Mean cross-validation score: 4.731459777248164
Standard deviation of cross-validation scores: 0.27036481240267596

data['medv_category'] = pd.cut(data['lstat'], bins=3, labels=["Low",
"Medium", "High"])

X = data.drop(['lstat', 'medv_category'], axis=1)
y = data['medv_category']

lda = LinearDiscriminantAnalysis()

cv_strategy = KFold(n_splits=5, shuffle=True, random_state=42)

cv_scores = cross_val_score(lda, X, y, cv=cv_strategy,
scoring='accuracy')

print(f"Cross-validation scores for each fold: {cv_scores}")
print(f"Mean cross-validation score: {cv_scores.mean()}")
print(f"Standard deviation of cross-validation scores:
{cv_scores.std()}")
```

```
Cross-validation scores for each fold: [0.92156863 0.89108911
0.9009901  0.9009901  0.89108911]
Mean cross-validation score: 0.9011454086585129
Standard deviation of cross-validation scores: 0.01113026897470978
```

```python
random_points = data.drop('lstat', axis=1).sample(n=5)

print(random_points)
```

```
        crim    zn  indus  chas    nox     rm   age     dis  rad
tax  ...  \
236  0.52058   0.0   6.20   1.0  0.507  6.631  76.5  4.1480  8.0
307.0  ...
37   0.08014   0.0   5.96   0.0  0.499  5.850  41.5  3.9342  5.0
279.0  ...
105  0.13262   0.0   8.56   0.0  0.520  5.851  96.7  2.1069  5.0
384.0  ...
84   0.05059   0.0   4.49   0.0  0.449  6.389  48.0  4.7794  3.0
247.0  ...
268  0.54050  20.0   3.97   0.0  0.575  7.470  52.6  2.8720  5.0
264.0  ...

     MEDV_category_High  medv_category_Low  medv_category_Medium  \
236                 0.0                1.0                   0.0
37                  0.0                1.0                   0.0
105                 0.0                0.0                   1.0
84                  0.0                1.0                   0.0
268                 0.0                1.0                   0.0

     medv_category_High  MEDV_category_Low  MEDV_category_Medium  \
236                 0.0                0.0                   1.0
37                  0.0                0.0                   1.0
105                 0.0                0.0                   1.0
84                  0.0                0.0                   1.0
268                 0.0                0.0                   0.0

     MEDV_category_High  MEDV_category_Medium  MEDV_category_High  \
236                 0.0                   0.0                 0.0
37                  0.0                   0.0                 0.0
105                 0.0                   1.0                 0.0
84                  0.0                   0.0                 0.0
268                 1.0                   0.0                 0.0

     medv_category
236            Low
37             Low
105         Medium
84             Low
268            Low
```

```
[5 rows x 34 columns]
```

Step 7: Deployment Phase

```
X_reg = data[['lstat']]
y_reg = data['medv']

X_train_reg, X_test_reg, y_train_reg, y_test_reg =
train_test_split(X_reg, y_reg, test_size=0.2, random_state=42)

regressor = LinearRegression()
regressor.fit(X_train_reg, y_train_reg)

data['MEDV_category'] = pd.cut(data['medv'], bins=[0, 15, 30,
max(data['medv'])], labels=['Low', 'Medium', 'High'])
X_clf = data[['lstat']]  # Using only LSTAT for prediction
y_clf = data['MEDV_category']

label_encoder = LabelEncoder()
y_clf_encoded = label_encoder.fit_transform(y_clf)

X_train_clf, X_test_clf, y_train_clf, y_test_clf =
train_test_split(X_clf, y_clf_encoded, test_size=0.2, random_state=42)

classifier = DecisionTreeClassifier(max_depth=3, random_state=42)
classifier.fit(X_train_clf, y_train_clf)

X_new = np.array([[5], [15], [25]])  # New LSTAT values

predicted_medv = regressor.predict(X_new)
print("Predicted MEDV values based on LSTAT:", predicted_medv)

predicted_categories = classifier.predict(X_new)
predicted_category_labels =
label_encoder.inverse_transform(predicted_categories)
print("Predicted MEDV categories based on LSTAT:",
predicted_category_labels)

y_pred_reg = regressor.predict(X_test_reg)
mse = mean_squared_error(y_test_reg, y_pred_reg)
print(f"Regression MSE: {mse}")

y_pred_clf = classifier.predict(X_test_clf)
accuracy = accuracy_score(y_test_clf, y_pred_clf)
print(f"Classification Accuracy: {accuracy}")

Predicted MEDV values based on LSTAT: [30.00429531 20.33898629
10.67367727]
Predicted MEDV categories based on LSTAT: ['High' 'Medium' 'Low']
```

```
Regression MSE: 33.51954917268489
Classification Accuracy: 0.8137254901960784
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/base.py:464: UserWarning: X does not have valid
feature names, but LinearRegression was fitted with feature names
  warnings.warn(
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages/sklearn/base.py:464: UserWarning: X does not have valid
feature names, but DecisionTreeClassifier was fitted with feature
names
  warnings.warn(
```

```
pip install -U notebook-as-pdf
```

```
Defaulting to user installation because normal site-packages is not
writeable
Looking in links: /usr/share/pip-wheels
Collecting notebook-as-pdf
  Obtaining dependency information for notebook-as-pdf from
https://files.pythonhosted.org/packages/be/aa/33c6dc40a09b01d77a657e95
461932463e4c061ba623e6bbc4f6ab15634d/notebook_as_pdf-0.5.0-py3-none-
any.whl.metadata
  Downloading notebook_as_pdf-0.5.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: nbconvert in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from notebook-as-
pdf) (6.5.4)
Collecting pyppeteer (from notebook-as-pdf)
  Obtaining dependency information for pyppeteer from
https://files.pythonhosted.org/packages/3d/ee/fb2757a38025421fd3844a0e
d0a230b78c9c04a66355024436cf3005a70c/pyppeteer-2.0.0-py3-none-
any.whl.metadata
  Downloading pyppeteer-2.0.0-py3-none-any.whl.metadata (7.1 kB)
Collecting PyPDF2 (from notebook-as-pdf)
  Obtaining dependency information for PyPDF2 from
https://files.pythonhosted.org/packages/8e/5e/c86a5643653825d3c913719e
788e41386bee415c2b87b4f955432f2de6b2/pypdf2-3.0.1-py3-none-
any.whl.metadata
  Downloading pypdf2-3.0.1-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: lxml in /opt/conda/envs/anaconda-panel-
2023.05-py310/lib/python3.11/site-packages (from nbconvert->notebook-
as-pdf) (4.9.3)
Requirement already satisfied: beautifulsoup4 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (4.12.2)
Requirement already satisfied: bleach in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from nbconvert-
>notebook-as-pdf) (4.1.0)
Requirement already satisfied: defusedxml in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from nbconvert-
```

```
>notebook-as-pdf) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (0.4)
Requirement already satisfied: jinja2>=3.0 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (3.1.2)
Requirement already satisfied: jupyter-core>=4.7 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (5.3.0)
Requirement already satisfied: jupyterlab-pygments in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (2.1.1)
Requirement already satisfied: mistune<2,>=0.8.1 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (0.5.13)
Requirement already satisfied: nbformat>=5.1 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (5.9.2)
Requirement already satisfied: packaging in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from nbconvert-
>notebook-as-pdf) (23.1)
Requirement already satisfied: pandocfilters>=1.4.1 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (2.15.1)
Requirement already satisfied: tinycss2 in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from nbconvert-
>notebook-as-pdf) (1.2.1)
Requirement already satisfied: traitlets>=5.0 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbconvert->notebook-as-pdf) (5.7.1)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from pyppeteer->notebook-as-pdf) (1.4.4)
Requirement already satisfied: certifi>=2023 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from pyppeteer->notebook-as-pdf) (2023.7.22)
Requirement already satisfied: importlib-metadata>=1.4 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from pyppeteer->notebook-as-pdf) (6.0.0)
```

```
Collecting pyee<12.0.0,>=11.0.0 (from pyppeteer->notebook-as-pdf)
  Obtaining dependency information for pyee<12.0.0,>=11.0.0 from
https://files.pythonhosted.org/packages/16/cc/5cea8a0a0d3deb90b5a0d39a
d1a6a1ccaa40a9ea86d793eb8a49d32a6ed0/pyee-11.1.0-py3-none-
any.whl.metadata
  Downloading pyee-11.1.0-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from pyppeteer->notebook-as-pdf) (4.65.0)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from pyppeteer->notebook-as-pdf) (1.26.16)
Collecting websockets<11.0,>=10.0 (from pyppeteer->notebook-as-pdf)
  Obtaining dependency information for websockets<11.0,>=10.0 from
https://files.pythonhosted.org/packages/d5/5d/d0b039f0db0bb1fea9343772
1cf3cd8a244ad02a86960c38a3853d5e1fab/websockets-10.4-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux
2014_x86_64.whl.metadata
  Downloading websockets-10.4-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux
2014_x86_64.whl.metadata (6.4 kB)
Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from importlib-
metadata>=1.4->pyppeteer->notebook-as-pdf) (3.11.0)
Requirement already satisfied: platformdirs>=2.5 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from jupyter-core>=4.7->nbconvert->notebook-as-pdf) (3.10.0)
Requirement already satisfied: jupyter-client>=6.1.5 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbclient>=0.5.0->nbconvert->notebook-as-pdf) (7.4.9)
Requirement already satisfied: nest-asyncio in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbclient>=0.5.0->nbconvert->notebook-as-pdf) (1.5.6)
Requirement already satisfied: fastjsonschema in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbformat>=5.1->nbconvert->notebook-as-pdf) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from nbformat>=5.1->nbconvert->notebook-as-pdf) (4.17.3)
Requirement already satisfied: typing-extensions in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from pyee<12.0.0,>=11.0.0->pyppeteer->notebook-as-pdf)
(4.7.1)
Requirement already satisfied: soupsieve>1.2 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from beautifulsoup4->nbconvert->notebook-as-pdf) (2.4)
Requirement already satisfied: six>=1.9.0 in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from bleach-
>nbconvert->notebook-as-pdf) (1.16.0)
```

```
Requirement already satisfied: webencodings in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from bleach->nbconvert->notebook-as-pdf) (0.5.1)
Requirement already satisfied: attrs>=17.4.0 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert->notebook-as-
pdf) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!
=0.17.2,>=0.14.0 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from jsonschema>=2.6->nbformat>=5.1->nbconvert->notebook-as-
pdf) (0.18.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert-
>notebook-as-pdf) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert-
>notebook-as-pdf) (23.2.0)
Requirement already satisfied: tornado>=6.2 in
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-
packages (from jupyter-client>=6.1.5->nbclient>=0.5.0->nbconvert-
>notebook-as-pdf) (6.3.2)
Downloading notebook_as_pdf-0.5.0-py3-none-any.whl (6.5 kB)
Downloading pypdf2-3.0.1-py3-none-any.whl (232 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 8.1 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82.9/82.9 kB 2.5 MB/s eta
0:00:00
anylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (107 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 107.4/107.4 kB 3.6 MB/s eta
0:00:00
  WARNING: The script pyppeteer-install is installed in
'/home/8aed5a14-1379-47cb-91b0-fe16e610da53/.local/bin' which is not
on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress
this warning, use --no-warn-script-location.
Successfully installed PyPDF2-3.0.1 notebook-as-pdf-0.5.0 pyee-11.1.0
pyppeteer-2.0.0 websockets-10.4
Note: you may need to restart the kernel to use updated packages.
```

```
pip pyppeteer-install
```

```
ERROR: unknown command "pyppeteer-install"
Note: you may need to restart the kernel to use updated packages.
```

BOSTON HOUSING DATASET ANALYSIS REPORT

4-29-2024

MATTHEW HENAO

Z23685608
Professor Juan Yepes

Overview

The Boston Housing Dataset is a well-known dataset used primarily in machine learning and statistics for predicting housing prices through regression analysis. It was originally compiled by the U.S. Census Service in the 1970s and has been extensively used to analyze and predict housing market behaviors.

Data Description

These features are both directly and indirectly related to the housing prices in the area. Below is a description of each feature included in the dataset:

- CRIM: Per capita crime rate by town.
- ZN: Proportion of residential land zoned for lots over 25,000 sq. ft.
- INDUS: Proportion of non-retail business acres per town.
- CHAS: Charles River dummy variable
- NOX: Nitric oxides concentration (parts per 10 million).
- RM: Average number of rooms per dwelling.
- AGE: Proportion of owner-occupied units built prior to 1940. (target variable)
- DIS: Weighted distances to five Boston employment centers.
- RAD: Index of accessibility to radial highways.
- TAX: Full-value property tax rate per $10,000.
- PTRATIO: Pupil-teacher ratio by town.
- B: where B is the proportion of black residents by town.
- LSTAT: Percentage of lower status of the population. (target variable)
- MEDV: Median value of owner-occupied homes in $1000s

VIF Analysis

Constant (VIF = 641.74): The high VIF for the constant term typically reflects the inclusion of other constant terms in the model or a high degree of multicollinearity among the variables.

CRIM (VIF = 1.83) and CHAS (VIF = 1.09): These variables exhibit low VIFs, indicating that they do not have strong linear relationships with other predictors in the model. This suggests that these variables provide unique information not captured by other variables.

ZN (VIF = 2.32), INDUS (VIF = 3.99), RM (VIF = 2.18), PTRATIO (VIF = 1.98), B (VIF = 1.38), and LSTAT (VIF = 3.24): These variables have moderate VIF values, suggesting moderate correlation with other variables but not enough to be overly concerning according to the VIF threshold of 10.

NOX (VIF = 4.28), DIS (VIF = 4.06), and MEDV (VIF = 3.86): These values are approaching the higher end of the moderate range. They indicate some level of multicollinearity, but still below the threshold that would typically cause alarm.

RAD (VIF = 7.76) and TAX (VIF = 9.20): These variables are approaching the threshold of 10, suggesting that they are highly correlated with other variables in the model.

Analysis of Linear Regression Model Performance

Coefficients & Intercept:

The coefficients derived from the model for most variables are extremely close to zero, with the exception of a significant coefficient for one of the variables (approximately 7.10). This suggests that most of the predictors have minimal influence on the model, except for this particular variable which appears to be a primary driver in predicting the target variable.

The model's intercept is approximately 12.46, indicating the average expected value of the dependent variable when all predictors are at their mean value.

Mean Squared Error (MSE):

The MSE is extremely low (approximately $2.15\times10-28$ 2.15×10^{-28}), indicating that the model predictions are almost exactly the same as the actual values. In practical scenarios, such a low MSE is unusual and might indicate overfitting or issues with the test dataset or model setup.

R-squared (R^2):

The R^2 value is 1.0, which means the model explains 100% of the variance in the dependent variable from the predictors. This is an exceptionally perfect score and is typically suspect in real-world data analyses as it suggests a potentially overfitted model.

Analysis of Classification Model Performance Using LDA

The model achieved an overall accuracy of 86.27%. This indicates that, on average, the model correctly predicts the class of an observation 86.27% of the time across all predictions made.

Detailed Classification Metrics

Class 0 (Precision: 1.00, Recall: 0.67, F1-Score: 0.80): The model perfectly identifies all relevant instances of Class 0 (precision = 1.00), but it only correctly identifies about 67% of actual instances of this class (recall = 0.67). This suggests some misclassification of Class 0 instances as belonging to other classes.

Class 1 (Precision: 0.80, Recall: 0.70, F1-Score: 0.74): For Class 1, the model is quite precise but not as robust in recall, indicating that while the predictions made are often correct, it misses about 30% of actual cases.

Class 2 (Precision: 0.68, Recall: 0.85, F1-Score: 0.76): Class 2 shows a lower precision but higher recall, meaning the model captures a good portion of Class 2 instances but also mislabels other classes as Class 2.

Class 3 (Precision: 0.96, Recall: 0.98, F1-Score: 0.97): The model performs excellently with Class 3, with both high precision and recall, indicating both accurate and comprehensive capture of Class 3 instances.

Analysis of Polynomial Regression Model Performance

Model Descriptions and Performance

3rd-degree Polynomial Regression (MSE: 36.52):

The model with polynomial features of degree 3 resulted in an MSE of 36.52. This indicates a moderate level of prediction error, suggesting that while the model can capture some non-linearity in the data, there might be room for improvement or a need for further tuning.

4th-degree Polynomial Regression (MSE: 19.68):

Increasing the degree to 4 significantly improves the model's performance, with the MSE reducing to 19.68. This improvement suggests that the additional complexity provided by the 4th-degree terms helps the model to better capture the underlying patterns in the data.

5th-degree Polynomial Regression (MSE: 35.87):

Surprisingly, further increasing the polynomial degree to 5 results in an MSE that is nearly as high as the 3rd-degree model, at 35.87. This could indicate overfitting, where the model becomes too tailored to the training data, losing its generalizability and thus performing poorly on new, unseen data.

Analysis of Decision Tree Regressor Performance on the MEDV Variable

Overall Accuracy: The classifier achieves an accuracy of 79.41%. This metric indicates a good general performance but hides class-specific details.

Class-specific Performance:

High (Precision: 1.00, Recall: 0.25, F1-Score: 0.40): Perfect precision indicates that all predictions of the 'High' category were correct, but the low recall shows that the model failed to identify most of the actual 'High' cases.

Low (Precision: 0.81, Recall: 0.89, F1-Score: 0.85): The model performs well for the 'Low' category, correctly identifying a high percentage of cases, though there are some false positives.

Medium (Precision: 0.74, Recall: 0.68, F1-Score: 0.71): This category sees balanced but moderate scores in both precision and recall, indicating a fair performance.

K-Means Clustering and Anomaly Detection Analysis

Cluster Profiling

Cluster 0: Characterized by relatively high values in features like 'nox', 'age', and 'tax', suggesting this cluster might consist of older areas with higher pollution and tax rates.

Cluster 1: Shows moderate negative values in 'indus' and 'nox' and positive values in 'zn' and 'rm', indicating this cluster likely represents residential areas with larger homes and less industrial activity.

Cluster 2: Marked by extreme negative values in several features and high values in 'MEDV_category_High', suggesting these areas are significantly different, potentially more affluent or economically distinct from others.

Feature Importance

The absolute values of the centroids indicate the relative importance of each feature in defining the cluster. Higher values indicate a stronger role in the cluster's profile. For instance, 'MEDV_category_High' has a dominant presence in Cluster 2, reflecting its unique demographic or economic status.

Cluster Validation Metrics

Silhouette Score (0.328): Indicates a fair separation between clusters, but there's room for improvement as values closer to 1 represent better-defined clusters.

Davies-Bouldin Score (1.025): A lower score (closer to 0) is better, suggesting that the clusters are reasonably compact and well-separated, although there could be some overlap.

Anomaly Detection

Detected Anomalies: A total of 21 data points were identified as anomalies based on their distance from the nearest cluster centroid. These are likely extreme or unusual cases within the dataset.

Anomalies Profile: These anomalies include properties with unusually high or low values in certain features such as 'crim', 'zn', 'indus', 'nox', and 'rm'. For example, index 388 shows extremely high crime rates and pollution levels, which significantly differ from most other data points.

Analysis of Linear Regression Performance with Cross-Validation

Cross-Validation Results

Scores for Each Fold: The Mean Squared Error (MSE) for each fold are as follows:

Fold 1: 4.580

Fold 2: 4.568

Fold 3: 5.107

Fold 4: 4.407

Fold 5: 4.995

These values represent the model's error metric in each cross-validation fold, indicating the average squared difference between the predicted and actual values.

Mean Cross-Validation Score: The mean MSE across all folds is approximately 4.731. This average indicates the model's typical performance in predicting 'LSTAT' across different subsets of the data, providing a robust measure of its predictive accuracy.

Standard Deviation of Cross-Validation Scores: The standard deviation of the MSE scores is about 0.270.