Milestone: - Exploration of Data Mining Models and Selection of Final Models

Uncovering Crime Patterns in Communities: A Data-Driven Approach

Group 87

Vishesh Gupta

Email Id: gupta.vishe@northeastern.edu

Signature: Vishesh Gupta

Submission: **March 3rd 2023**

Model Exploration:

Model exploration is the process of understanding how a trained machine learning model makes predictions and how it represents the relationships between the input features and the target variable

In the context of model exploration, we are performing supervised learning which can be useful because it provides a clear target variable to predict and evaluate the performance of the model.
In this project of Communities and Crime, we have a target variable called violent_crime_per_population, and our goal is to perform supervised learning on this variable using regression techniques. Regression is a type of supervised learning that aims to predict a continuous numerical value, such as the crime rate in this case.

To explore and understand the relationships between the input features and the target variable, we have used techniques such as visualization, Correlation, model interpretation, and analysis. For example, we have created pair plots and heat maps to visualize the relationships between different features and the target variable and used Dimension Reduction Techniques such as Principal component analysis to understand which features are capturing the highest number of variance which is really most important for the regression model.

To perform model exploration and selection on the Communities and Crime dataset, we first checked the values in our PCA dataframe after standardizing it. We then added the target variable, violent_crime_rate_per_population, to the PCA dataframe to perform model exploration. To do this, we removed the same variable from our standardized dataset to obtain separate input and output sets. For model performance, we split our data into training and testing sets in a ratio of 80:20 for both the input predictor variables and the output target variable. This was done using the following code snippet:

In [76]:
```
##Train test split
import pandas as pd
from sklearn.model_selection import train_test_split


# Shuffle the dataset
shuffled_dataset = model_data.sample(frac=1, random_state=42)

# Split the dataset into training, and testing sets
train_data, test_data = train_test_split(shuffled_dataset, test_size=0.2, random_state=42)
#train_data, validation_data = train_test_split(train_data, test_size=0.25, random_state=42)

print(train_data.shape)
print(test_data.shape)
```

```
(1595, 16)
(399, 16)
```

Here, X represents the standardized input predictor variables (after PCA), and y represents the target variable (violent_crime_per_population). We split the data into a training set (80%), a testing set (20%) using the train_test_split() function from scikit-learn.

```
In [81]: train_X, train_y = train_data.drop("ViolentCrimesPerPop", axis=1), train_data["ViolentCrimesPerPop"]
         test_X, test_y = test_data.drop("ViolentCrimesPerPop", axis=1), test_data["ViolentCrimesPerPop"]

         print(str(train_X.shape) + " - " + str(train_y.shape))
         print(str(test_X.shape) + " - " + str(test_y.shape))

         (1595, 15) - (1595,)
         (399, 15) - (399,)
```

We can then use the training set to fit different regression models. Finally, we can evaluate the performance of the selected model on the testing set to estimate its generalization performance.

Model Selection:

In terms of Machine Learning: Model selection is an important step in machine learning and involves choosing the best model or algorithm to use for a given problem. In the case of the Communities and Crime dataset, since we are performing regression, we need to choose a regression model that can accurately predict crime rates in different communities based on a set of input features.

As an illustration, we could aim to identify the suitable hyperparameters for a specific machine learning algorithm. These hyperparameters are the parameters of the learning method that must be set prior to the model fitting.

There are many different regression models to choose from, including linear regression, decision trees, random forests, support vector regression, and neural networks, among others. The best model for the task depends on a number of factors, including the size and complexity of the dataset, the number of input features, the quality of the data, and the desired level of accuracy.

In this milestone, our primary goal is to evaluate the performance of different regression models on our training, validation and testing data. We will achieve this by applying six different regression models to the standardized input data and comparing their performance metrics. The models we will be using include linear regression, Lasso regression, Ridge regression, ElasticNet, Decision Tree regression, and Random Forest regression.

We will first fit each model to the training data, using the training set split that was created in the previous step. We will then use the trained models to make predictions on the testing set and evaluate their performance using several metrics such as mean squared error (MSE), root mean squared error (RMSE), and coefficient of determination (R^2).

The goal of this evaluation process is to identify which regression model performs the best on our dataset and select it as our final model. By comparing the performance of different models, we can gain insights into the strengths and weaknesses of each model and select the one that provides the best balance of accuracy and interpretability.

Step1) The first regression model we have used in this project is linear regression. Linear regression is a simple yet powerful technique that is widely used in machine learning for predicting numerical values, such as crime rates in different communities.

It is important because it is simple, interpretable, and computationally efficient, serving as a baseline for comparing more complex models. It assumes a linear relationship between input features and the target variable.

In [82]:
```
## Linear Regression

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Instantiate the linear regression model
lr = LinearRegression()

# Fit the model to the training data
lr.fit(train_X, train_y)

# Make predictions on the test set
y_pred_test_lr = lr.predict(test_X)

# Calculate the mean squared error on the test set
mse_test_lr = mean_squared_error(test_y, y_pred_test_lr)

# Print the mean squared error on the test set
print("Mean Squared Error on test set: ", mse_test_lr)
```

```
Mean Squared Error on test set:  0.017870786515017806
```

Step2) The second regression model we have used in this project is Lasso regression. Lasso regression is a type of linear regression that uses regularization to prevent overfitting and improve the generalization performance of the model. Furthermore, Lasso regression can be used to perform feature selection and reduce the dimensionality of the input data, which can improve the computational efficiency of the model and reduce the risk of overfitting.

In [68]:
```
## Lasso

regressor = Lasso(alpha=0.1)
regressor.fit(train_X, train_y)

test_y_pred = regressor.predict(test_X)

mse = mean_squared_error(test_y, test_y_pred)
mae = mean_absolute_error(test_y, test_y_pred)
medae = median_absolute_error(test_y, test_y_pred)
r2 = r2_score(test_y, test_y_pred)

print(mse)
print(mae)
print(medae)
print(r2)
```

```
0.027391123111096972
0.11968206772619135
0.08831187523571964
0.49637335780795344
```

Step-3) The third regression model we have used in this project is Ridge regression. Ridge regression is a type of linear regression that also uses regularization to prevent overfitting and improve the generalization performance of the model. Furthermore, Ridge regression can also be used to perform feature selection and reduce the dimensionality of the input data, similar to Lasso regression.

Ridge regression is important to use because it can help us reduce the impact of input features that are not relevant for predicting the target variable. This is done by penalizing the input coefficients that have large magnitudes, which can lead to a more robust and accurate model.
In summary, Ridge regression is an important model selection technique that can help us reduce the impact of irrelevant input features and prevent overfitting.

In [67]:
```
## Ridge Regression

from sklearn.linear_model import Ridge

regressor = Ridge(alpha=0.5)
regressor.fit(train_X, train_y)

test_y_pred = regressor.predict(test_X)

mse = mean_squared_error(test_y, test_y_pred)
mae = mean_absolute_error(test_y, test_y_pred)
medae = median_absolute_error(test_y, test_y_pred)
r2 = r2_score(test_y, test_y_pred)

print(mse)
print(mae)
print(medae)
print(r2)
```

```
0.01787597064183314
0.09671482470497135
0.06476588214317221
0.6713236243086829
```

Step-4) The fourth regression model used in this project is Random Forest regression. Random Forest is an ensemble learning method that combines multiple Decision Trees to improve the prediction accuracy and reduce overfitting. Random Forest regression is important here to use because it can capture complex nonlinear relationships and interactions between input features, similar to Decision Trees. However, by combining multiple trees, it can improve the generalization performance of the model and reduce the variance.

Furthermore, Random Forest can also be used for feature selection and data preprocessing, similar to Decision Trees.

In [71]:
```
## Random Forest

regressor = RandomForestRegressor(n_estimators=100, max_depth=3)
regressor.fit(train_X, train_y)

test_y_pred = regressor.predict(test_X)

mse = mean_squared_error(test_y, test_y_pred)
mae = mean_absolute_error(test_y, test_y_pred)
medae = median_absolute_error(test_y, test_y_pred)
r2 = r2_score(test_y, test_y_pred)

print(mse)
print(mae)
print(medae)
print(r2)
```

```
0.023680362948136986
0.11277242818804184
0.07985259754466487
0.5646012166391391
```

Step-5) The fifth regression model used in this project is Elastic Net regression. Elastic Net is a type of linear regression that combines both Lasso and Ridge regularization to improve the generalization performance of the model.

In [69]:
```
## Elastic

regressor = ElasticNet(alpha=0.1, l1_ratio=0.5)
regressor.fit(train_X, train_y)

test_y_pred = regressor.predict(test_X)

mse = mean_squared_error(test_y, test_y_pred)
mae = mean_absolute_error(test_y, test_y_pred)
medae = median_absolute_error(test_y, test_y_pred)
r2 = r2_score(test_y, test_y_pred)

print(mse)
print(mae)
print(medae)
print(r2)
```

```
0.022136283135636634
0.10712089179201809
0.08119226123621218
0.5929914264195855
```

Elastic Net regression is important here to use because it can handle datasets with many input features and prevent overfitting by penalizing input coefficients with large magnitudes, similar to Ridge and Lasso regression. However, by combining both regularization methods, it can achieve better performance than using only one method.

Furthermore, Elastic Net can also be used for feature selection and reduce dimensionality, similar to Ridge and Lasso regression.

Till now we have used the below regression models:

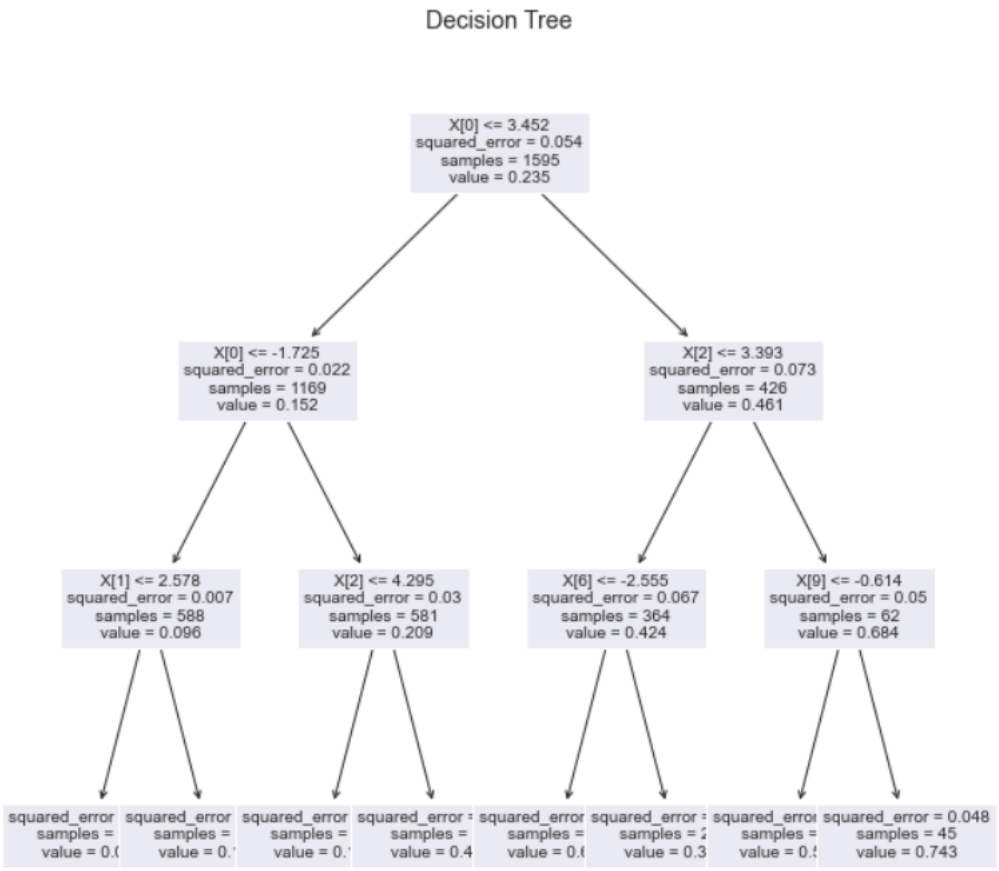
1. Linear Regression
2. Lasso Regression
3. Ridge Regression
4. Elastic Net Regression.
5. Random Forest Regression

Step-6) In this project, we have incorporated Decision Tree regression as one of the six regression models used to predict the target variable, violent crime rate per population, based on input predictor variables. Decision Tree regression is an important model selection technique because it can capture complex nonlinear relationships, is easy to understand and interpret, and can handle missing data and outliers effectively. It also provides insights into the most informative input features for the final model.

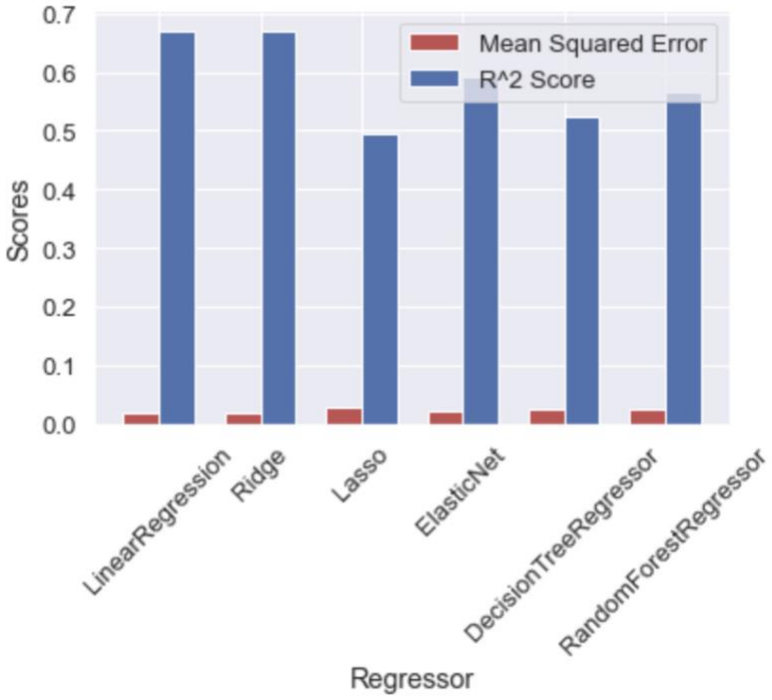
In [70]:

```
## Decision tree regressor

regressor = DecisionTreeRegressor(max_depth=3)
regressor.fit(train_X, train_y)

test_y_pred = regressor.predict(test_X)

mse = mean_squared_error(test_y, test_y_pred)
mae = mean_absolute_error(test_y, test_y_pred)
medae = median_absolute_error(test_y, test_y_pred)
r2 = r2_score(test_y, test_y_pred)

print(mse)
print(mae)
print(medae)
print(r2)
```

```
0.026359152500176692
0.1194180444171097
0.0857011070110702
0.5153476762946642
```



Therefore, Ridge regression is an essential model to incorporate in our analysis to ensure that we are exploring all possible model options and selecting the best model for our dataset.



We can also use cross-validation, hyperparameter tuning, and model selection techniques to build the best possible regression model for the Communities and Crime dataset. For future reference, we can also perform classification tasks on this dataset, but as of now, our main goal is to perform regression. By using supervised learning techniques, we gained insights into the factors that contribute to crime rates and develop models that can accurately predict crime rates in different communities.

Overall, this approach allows us to perform model exploration and selection in a systematic and controlled way, and helps us to identify the best regression model for predicting crime rates in different communities based on a set of input features.