Project Stage 3

Prepared By:
Yining Wang SID: 530023821
Danica Zhang SID: 510644297
Emily Lin SID: 530508591
Jack Li SID: 530511063

[10.29.2023]

TABLE OF CONTENTS

Introduction

Dataset Description:

The dataset used in this analysis explores the relationship between socioeconomic factors and crime rates in the United States from 2013 to 2019. It encompasses information on various states, focusing on essential attributes such as income, unemployment rates, education attainment, and crime rates.

Attribute to Predict:

In this analysis, the main attribute of interest is crime rates–a quantitative variable. To obtain predictions on the values of crime rates, machine-learning models were selectively chosen, with their outputs evaluated to determine the extent of their accuracy. Observing the models additionally allows for a better understanding of the fluctuations and variations in the numbers under the influence of other socioeconomic factors.

Data Splitting:

75% of the data was used to train the predictive model, while the remaining 25% was reserved for evaluating the model's performance. The purpose of this division is to ensure that the model can learn from a significant portion of the data while being rigorously tested on unseen data to assess its predictive capabilities and generalization.

Aims:

By identifying the key drivers of crime, we aim at uncovering patterns and trends that can inform crime prevention strategies and policy making. Law enforcement agencies, government and policymakers can learn from our predictions and allocate their resources more effectively while developing targeted strategies for crime prevention. The general public can also benefit from our report by gaining a clearer understanding of the factors that influence crime rates in their communities.

INDIVIDUAL 1 - Yining Wang 530023821

```
df=pd.read_excel("Merged_Dataset.xlsx")
df.rename(columns={"Rates of all crimes": "Crime rates (per 100,000 population)",
                   "Real personal income": "Real personal income (million dollars)",
                   "Unemployment_rate": "Unemployment_rate (%)",
                   "Percentage with high school completion or higher (%)  total":
                   "Percentage of population who completed high school (%)"},
                    inplace=True)
#Rates are the number of reported offenses per 100,000 population
#Real personal income is in the unit of million dollars
#Unemployment rate is the actual percentage
#Percentage with high school completion or higher is the % of population who completed high school
df
```

Producing Predictive Models

- Data Preprocessing: The dataset underwent careful preprocessing in stage 1 where any missing or incorrect values were addressed, guaranteeing that the dataset was of high quality and free from errors. Meanwhile, for clarity and interpretability, we used ***`df.rename`*** to **rename specific columns, and appending brackets to denote the respective units of measurement.** "Real personal income (million dollars)," "Unemployment rate (%)," and "Percentage of the population who completed high school (%)" were chosen as our independent variables and "Crime rates (per 100,000 population)" as the target variable to predict.

- Train/Test Splitting: Using ***`train_test_split`***, ***`test_size=0.25`*** specifies that the dataset is split into training and testing sets with a **75/25 split ratio**. Such an approach is reasonable as the model has access to a substantial amount of training data (75%) to learn and build its predictive capability while adequate testing data (25%) is reserved for us to accurately assess our model's performance. ***`random_state=42`*** is used to seed the random number generator, ensuring that both training and testing sets are randomly selected and are representative of the entire dataset.

- Model Selection and Explanation: Random Forest is a combination of multiple decision trees. Each decision tree is constructed independently using the Bootstrap Aggregating technique (a random sample of data in a training set is selected with replacement) on different subsets of the training data, which greatly improves the performance and accuracy of the algorithm. The final prediction is obtained by averaging the results from individual trees in the random forest. **As crime rates fluctuate frequently and are impacted by various factors, we expect a non-linear relationship between socioeconomic factors and crime rates. Therefore, It is a reasonable approach to employ a random forest model since it mitigates overfitting and enhances prediction accuracy by considering multiple trees rather than relying on a single tree's predictions. Meanwhile, the nature of Random Forest makes it resilient to outliers and noisy data, accommodating the complexity and variability inherent in predicting crime rates.**

- Training and Fitting: Using ***`RandomForestRegressor,`*** we create a Random Forest regression model with ***`(n_estimators=100)`*** specifies that our model was fitted with 100 decision trees and the final prediction is an average or majority vote from all these trees. ***`rf_regr.fit(X_train, y_train)`*** shows that the ***`.fit()`*** method is used to train our random forest model, as ***`y_pred_rf = rf_regr.predict(X_test)`*** presented.

```
# Load data into a DataFrame
X = df[['Real personal income (million dollors)',
        'Unemployment_rate (%)',
        'Percentage of population who completed high school (%)']]
y = df['Crime rates (per 100,000 population)']

# Split the data into training and testing sets(0.75/0.25)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Create and fit a Random Forest regression model
rf_regr = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regr.fit(X_train, y_train)

# Predict the values
y_pred_rf = rf_regr.predict(X_test)
```

Evaluations of Predictive Models

Measurement: To assess the model's performance, two key metrics and plotting methods were used:

```
# Calculate RMSE for Random Forest
rmse_rf = metrics.mean_squared_error(y_test, y_pred_rf, squared=False)

# Calculate R-squared for Random Forest
r_squared_rf = rf_regr.score(X_test, y_test)
```

Root Mean Squared Error (RMSE): RMSE measures the square root of the average squared differences between actual and predicted values and is calculated using ***metrics.mean_squared_error***, quantifying the prediction error. Smaller RMSE values indicate better model performance, as they mean the model's predictions are closer to the actual values on average. In our specific case, the Random Forest model's RMSE is calculated to be 513.032. **While RMSE of this magnitude may initially appear large, it's crucial to look at it within the scope of the problem we are addressing. Crime rates, which fluctuate between 0 and 5000 per 100,000 population, encompass a broad range of values and exhibit substantial variability. Given this, an RMSE of 513.032 can be deemed acceptable.**

R-squared (R^2) Score: R-squared indicates the percentage of variance in the target variable that can be explained by the independent variables. A model with an r square greater than 0.5 is generally considered a moderate to good fit. In our case, ***rf_regr.score*** returned a r square of 0.512, **This result signifies that our Random Forest model is capable of explaining approximately 51.2% of the variance observed in crime rates. Therefore, we can reasonably conclude that our model is a moderate fit for the dataset.** The choice of Random Forest effectively captures complex, non-linear relationships.

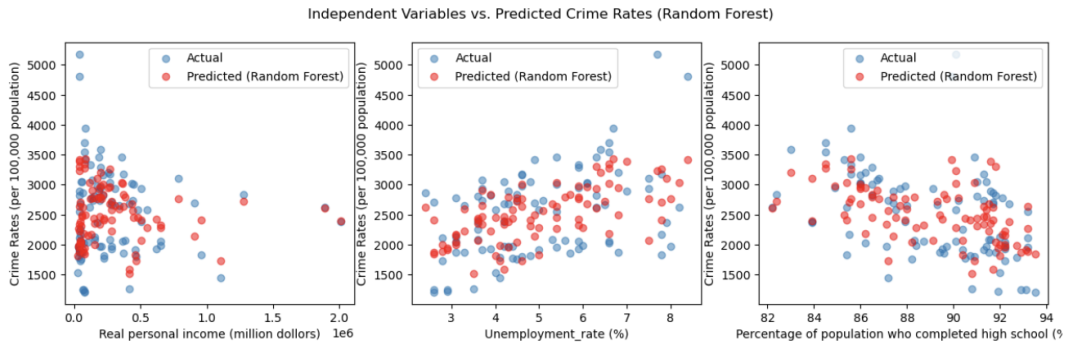
```
# Create subplots for each independent variable
fig, axs = plt.subplots(1, 3, figsize=(15, 4))

# Plot each independent variable against predicted crime rates
for i, col in enumerate(X.columns):
    axs[i].scatter(X_test[col], y_test, label='Actual', alpha=0.5)
    axs[i].scatter(X_test[col], y_pred_rf, label='Predicted (Random Forest)', color='red', alpha=0.5)
    axs[i].set_xlabel(col)
    axs[i].set_ylabel("Crime Rates (per 100,000 population)")
    axs[i].legend()

# Add a title to the entire figure
plt.suptitle("Independent Variables vs. Predicted Crime Rates (Random Forest)")

# Show the plot
plt.show()
```

Plotting: We created subplots for each independent variable with actual crime rates (in blue) against predicted crime rates(in red). The closer the predicted value(in red) to the actual value(in blue), the better the model. **From observations, it is evident that the model doesn't make exact predictions in every instance. However, the predicted data points still exhibit some degree of proximity to the actual results.** This proximity highlights the effectiveness of the Random Forest model in capturing the intricate patterns embedded in the data and making reliable predictions. **No overfitting or underfitting is observed throughout the measurement.**



However, **it is also important to look at the nature of the training method and the dataset to determine whether the model is a good fit. One notable characteristic of the dataset is the presence of multicollinearity among the three independent variables**: income, unemployment rate, and education attainment. This high correlation among independent variables can impact the quality and accuracy of predictions, posing challenges for the random forest model to distinguish the unique contribution of each independent variable. **Another critical characteristic of the dataset is its relatively small size**, with fewer than 300 sets of values used for training and testing. The relatively limited size of the dataset is an inherent limitation that may introduce some variability in random forest predictions.

INDIVIDUAL 2 - SID: 530508591

Similarly to Stages 1 and 2, we began by importing essential libraries for data manipulation, numerical calculations, visualization, and modeling. We worked with the same dataset, which contains entries related to potential factors influencing crime rates in U.S. states across the years 2013-2019. The predictive model produced here involves two independent features, 'Disposable personal income' and 'Percentage with bachelor's or higher degree (%) total,' and one dependent variable (target), 'Rates of all crimes,' denoted as 'X' and 'y,' respectively.

```
# Extract the independent variables (features) and the target variable
X = data[['Disposable personal income', "Percentage with bachelor's or higher degree (%)  total"]]
y = data['Rates of all crimes']
```

To evaluate our model's performance on new, unseen data, we divided the selected features into training and testing sets using scikit-learn's 'train_test_split' function. However, before performing the split, we ensured the number of outliers were minimized by specifying a Z-score threshold of 2 that essentially filters the dataset to a 95% confidence interval in a standard normal distribution. The resulting data which fall within the range are named 'X_no_outliers' and 'y_no_outliers'.

```
# Define a Z-Score threshold (e.g., 2 or -2 for a 95% confidence interval)
z_threshold = 2

# Remove outliers using Z-Score
z_scores = np.abs(stats.zscore(X))
X_no_outliers = X[(z_scores < z_threshold).all(axis=1)]
y_no_outliers = y[(z_scores < z_threshold).all(axis=1)]
```

We allocated 25% of the dataset to the testing set while using 75% for training, as shown in the parameter ‘test_size’. Optionally, we set a specific random seed with 'random_state=42' for reproducibility. The outputs of the split are labeled as 'X_train,' 'y_train,' and 'X_test,' 'y_test.'

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_no_outliers, y_no_outliers, test_size=0.25, random_state=42)
```

We generated a new set of polynomial features by applying ‘poly.fit_transform’ to both the training data and testing data. In our context, this approach proves highly advantageous because crime rates exhibit significant variability, largely influenced by external factors. This leads us to the conclusion that the relationship between the independent features and the target variable is not characterized by a straightforward linear association but, instead, demonstrates complex patterns. The degree chosen for the polynomial was adjusted to 5 after comparing the outcomes for the model’s performance, resulting in a more accurate curve fit for the model.

```
# Create polynomial features
poly = PolynomialFeatures(degree=5)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)
```

After the polynomial transformation, we built a Ridge regression model from the training data, with a specified regularization strength (‘alpha’) of 1.0. This hyperparameter controls the importance given to the regularization term, with larger values shrinking model coefficients towards zero and thus reducing model complexity. The model is used to make predictions on the transformed test data, output named as ‘y_pred’.

```
# Create and fit the Ridge regression model
alpha = 1  # You can adjust the regularization strength
model = Ridge(alpha=alpha)
model.fit(X_train_poly, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test_poly)
```

For a quantitative assessment of the model’s predictive accuracy, we calculated the Mean Squared Error (MSE) and R-squared scores. MSE quantifies the average squared difference between model predictions ('y_pred') and actual values ('y_test'), with lower MSE indicating better performance. The R2 score measures how effectively the model's independent features explain variance in the target, ranging from 0 to 1. However, since a higher degree polynomial was chosen, the MSE and R2 score was computed for both the training and testing set. Making a comparison between the respective values ensures the predictive model does not overfit the training data to a large extent (i.e. much larger R2 or greatly smaller MSE).

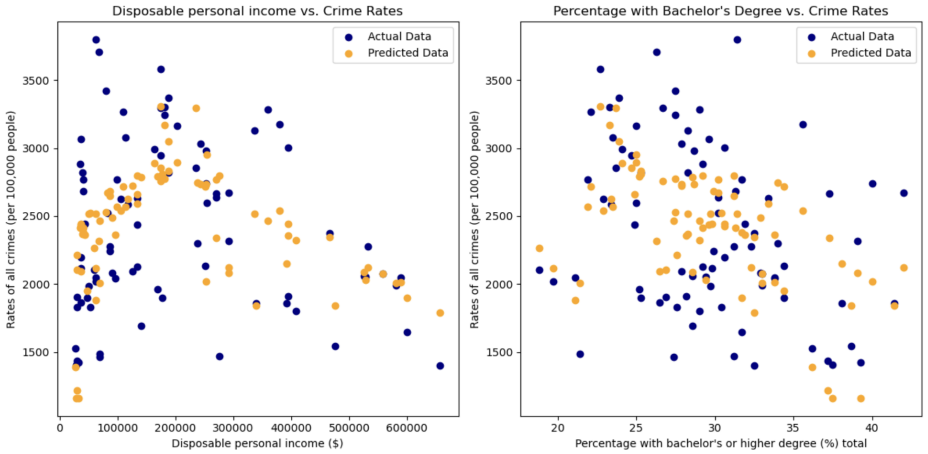
```
# Make predictions on the test data
y_pred = model.predict(X_test_poly)

# Calculate model performance metrics
mse_test = mean_squared_error(y_test, y_pred)
r2_test = r2_score(y_test, y_pred)

# Calculate MSE for training data
y_train_pred = model.predict(X_train_poly)
mse_train = mean_squared_error(y_train, y_train_pred)
r2_train = r2_score(y_train, y_train_pred)
```

```
Testing Mean Squared Error: 245796.03
Testing R-squared (R2) Score: 0.35
Training Mean Squared Error: 217121.78
Training R-squared (R2) Score: 0.42
```

Lastly, two subplots were constructed for better visualization purposes, both of which are valuable for comparing and assessing the accuracy of the model's predictions with the actual data. The first subplot displays the relationship between Disposable Personal Income and Crime Rates, whereas the second sets the independent variable to be the Percentage with Bachelor's of Higher degree (%) total and also plots it against Crime Rates. The alignment between the actual data points (coloured navy) and the predicted (coloured orange)



Additional justification for decision choices:

- The Ridge regression was specifically chosen to prevent overfitting, particularly beneficial in our case where our model includes polynomial features up to the 5th degree and given our dataset is of a relatively small size of (roughly 500 data values). The regularization technique allows the model to capture non-linear relationships between our independent features and target where multicollinearity exists.
- The 'alpha' parameter for regularization strength tuning is adjusted to fit the data as best as possible and also prevent overfitting. The value is altered in accordance with the output Mean squared error and R-squared scores, where a lower MSE and higher R2 score is desirable.
- Z-score threshold allows the model to be less influenced by extremities and can provide more reliable and accurate predictions. A 95% confidence interval is chosen as a conservative estimate, ensuring there is still the presence of uncertainty in the estimation but in a narrow range.
- No normalization or standardization was done on the data after trialing and validating our model's performance with and without scaling (by examining change in MSE and R2 scores). Original feature values were kept for better interpretability, making it easier to explain the impact of features on the model's predictions.

Evaluation of predictive model:

- Overall, the chosen Ridge regression model is moderate but definitely not optimal. The R2 scores for the training data sets reveals that the model explains around 35% of the variance in Crime Rates, compared to 42% on the training data.
- The MSE values returned for both the training and testing data sets are relatively high (up in the hundred thousands) but is acceptable to some degree given our chosen dataset uses values including very large values too (especially for Disposable Personal income which has values above 600000).
- However, the initial distribution of the actual data points shown in the scatter plot fail to exhibit a notable relationship, further explaining the poor performance of the Polynomial Regression model. Such variation in the data values make it harder for a model to be used for predictions and can only be accurate to a limited extent.

Individual 3 - SID: 510644297 (Decision Trees Regressor)

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error as MSE
from sklearn.model_selection import cross_val_score

✓ 0.0s

df = pd.read_excel('/Users/mac/Downloads/Merged Dataset.xlsx')
print(df.dtypes)

df = pd.DataFrame(df)
print(df.columns)
```

Producing Predictive Models

- Data Preprocessing: The dataset addressed all missing or incorrect values in stage 1. The dataset is clean and has a high quality.
- Train/Test Splitting: The dataset is split into training and testing sets by using *train_test_split*. The property of training and testing sets is a 75/25 ratio. This approach can assess and improve the model's performance. This step is crucial for evaluating how well the model generalizes to unseen data. *random_state = 42* is used to set a parameter that controls the randomization within the model. It equals 42 to ensure reproducibility.
- Model Selection and Explanation: Decision tree for regression is a model to solve regression problems for continuous target variables by capturing a non-linear trend. It recursively partitions the dataset into subsets based on the most informative features, eventually leading to predictions. The regression tree is trained on a dataset, and the impurity of a node is measured using the mean-squared error of the targets in that node. Three scatter plots show the relationships between crime rates and the other three variables, including "percentage with bachelor's or higher degree total", "total employment (numbers of jobs)", and "Real GDP", which are not linear. Decision tree is highly interpretable. It is clear and easily understandable to make predictions. Secondly, it is relatively unaffected by outliers in the dataset because it can split data based on order rather than the actual values. Thirdly, this approach is easy to implement and requires tuning fewer hyperparameters. Thus, it is a reasonable approach to consider for the dataset.

```
# Split data iinto 75% training and 25% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=21)

# Instantiate a DecisionTreeRegressor 'dt'
dt = DecisionTreeRegressor(max_depth = 4, min_samples_leaf = 3, random_state=21)
```

- Training and fitting: Using *DecisionTreeRegressor*, we make a decision tree regressor model. *max_depth = 4* means that the tree has a maximum of 4 decision nodes before it reaches the leaf nodes. The depth of the tree is limited to 4 to prevent overfitting. *min_samples_leaf = 0.1* sets the minimum number of samples in a leaf node of the decision tree. *fit()* is used to train the decision tree regressor model. Finally, we predict training and test sets separately.

Evaluations of Predictive Models

```
# Fit 'dt' to the training-set
dt.fit(X_train, y_train)

# Predict test-set labels
y_pred = dt.predict(X_test)

# Compute test-set MSE
mse_dt = MSE(y_test, y_pred)

# Compute test-set RMSE
rmse_dt = mse_dt**(1/2)

# rmse_dt
print(rmse_dt)

# Calculate R-squared for decisioin tree
r2_dt = dt.score(X_test, y_test)
print(r2_dt)

✓ 0.0s

433.1562335298421
0.6145762621497108
```

- Root Mean Squared Error (RMSE): RMSE measures the square root of the average squared differences between actual and predicted values. A smaller RMSE value indicates better model performance. The RMSE for test set is 433.1562335298421. The RMSE appears large, however, it can be accepted. There are two reasons. The first reason is that crime rates vary between 0 and 5000 per 100,000 population. The RMSE reflects the inherent variability in the dataset. Secondly, the dataset is limited and its size is small. The dataset is a combination of different datasets. Collection methods of different datasets may be different, therefore, bias may appear.

- R-squared Score: R-squared score is used to evaluate the goodness of fit of a model. It indicates the changes in the target variable based on the independent variables (features). The R-squared score is a value between 0 and 1. In the decision tree regressor model, it equals 0.6146. Thus, the decision tree regressor model is a moderate fit for the dataset.

```
# Diagnose bias and variance problems

# Evaluate the list of MSE ontained by 10-fold CV
# Set n_jobs to -1 in order to exploit all CPU cores in computation
MSE_CV = - cross_val_score(dt, X_train, y_train, cv=10, scoring='neg_mean_squared_error', n_jobs=-1)

# Fit 'dt' to the training set
dt.fit(X_train, y_train)
# Predict the labels of training set
y_predict_train = dt.predict(X_train)
# Predict the labels of test set
y_predict_test = dt.predict(X_test)

# CV MSE
print('CV MSE: {:.2f}'.format(MSE_CV.mean()))

# Training set MSE
print('Train MSE: {:.2f}'.format(MSE(y_train, y_predict_train)))
# Test set MSE
print('Test MSE: {:.2f}'.format(MSE(y_test, y_predict_test)))

#dt is overfitting and has a high variance
#dt suffers from high bias because MSE_CV and MSE_train are about equal and greater than MSE_test
#dt is overfitting because MSE_train << MSE_test
```

✓ 5.3s

```
CV MSE: 404105.33
Train MSE: 383046.45
Test MSE: 463032.63
```

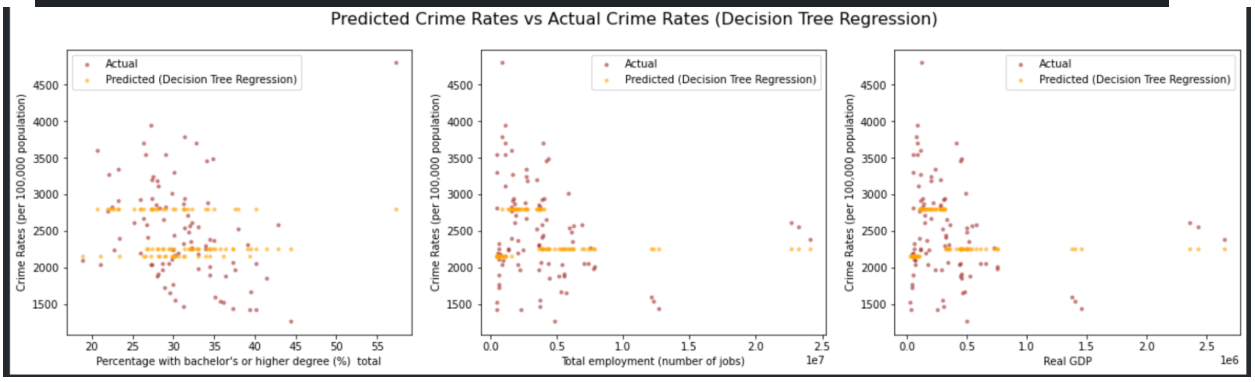
- Diagnose Bias and Variance Problems: We calculate mean squared errors for CV, train, and test sets. CV MSE is 404105.33. Train MSE is 383046.45. Test MSE is 463032.63. CV MSE is larger than train MSE. It means that the model is overfitting and has a high variance.

- Plotting: We create subplots for each independent variable and plot each independent variable compared with crime rates, then, display the graphs. For these three graphs, they show the model doesn't make accurate predictions. However, the model predicts some possible relationships to the actual results. When features increase, the number of points on the graphs decrease relatively.

```
# Create subplots for each independent variable
fig, axs = plt.subplots(1, 3, figsize=(20, 5))

# Plot each independent variable vs predited crime rates
for i, col in enumerate(X.columns):
    axs[i].scatter(X_test[col], y_test, marker='.', alpha = 0.5, label = 'Actual', color = 'brown')
    axs[i].scatter(X_test[col], y_predict_test, marker='.', alpha = 0.5, label = 'Predicted (Decision Tree Regression)',
                   color = 'orange')
    axs[i].set_xlabel(col)
    axs[i].set_ylabel("Crime Rates (per 100,000 population)")
    axs[i].legend()

# Add a title to the entire graphs
plt.suptitle('Predicted Crime Rates vs Actual Crime Rates (Decision Tree Regression)', fontsize=16)
```



Individual 4 (530511063)

Initially, the data was loaded in, and sorted and extracted into relevant columns. In this case, the features being “GDP” and “Per capita personal income” which are dependent variables, while “Crime rate (per 100,000 population)” is the independent variable, or in other words, the ‘target’.

Using ‘train_test_split’, the data is split into a training set and a test set, with the training set being 80% of the data and the test set being 20% of the data. Furthermore, the “random_state” parameter is set to 40 for ensuring reproducibility.

The degree of the polynomial regression is set to 4 which means that 4th degree polynomial features are being created. A PolynomialFeatures object is then created and applied to both the training set and the test set. By creating

To train the model, a Linear Regression model is created using “LinearRegression()”. This model is then trained with the training data with polynomial features. This fits a linear relationship between the transformed features and the target variable.

Finally, to make predictions and the two evaluation metrics.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv('/content/MergedDataset.csv')

# Assuming your data is stored in a DataFrame df
X = df[['Per capita personal income',
        'Percentage with high school completion or higher (%)  total', ]]
y = df['Rates of all crimes']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 40)

# Create polynomial features
degree = 4
poly = PolynomialFeatures(degree = degree)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Train a linear regression model with polynomial features
model = LinearRegression()
model.fit(X_train_poly, y_train)

# Make predictions
y_pred = model.predict(X_test_poly)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 450252.9643379293
R-squared: 0.1999403812584577
```

Evaluation

The high MSE suggests that the model's predictions are not very close to the actual values.

On the other hand, the relatively low r-squared value accounts for roughly 20% of the variance in the target variable, which implies that the model doesn't fit the data very well.

Based on the provided MSE and R^2 values, the model does not perform very effectively. The MSE is relatively high, indicating significant errors in predictions, and the R^2 value is low, suggesting that the model does not explain much of the variance in the data

Random Forest (Yining Wang–Individual 1)

Overview: Random Forest is an ensemble of decision trees. It builds multiple decision trees during the training phase and combines their results to produce more accurate and stable predictions. Random Forest effectively handles non-linear relationships in the data, making it suitable for our research question where we expect complex interactions between socioeconomic factors and crime rates.

Strengths: It is less impacted by the presence of multicollinearity among the independent variables and performs well in distinguishing the unique contributions of each variable. This can be proven by previous evaluation where random Forest shows a moderate R-squared score (0.512), explaining over 51% of the variance in crime rates. The model also mitigates the effects of overfitting, allowing it to provide consistent and dependable predictions.

Weaknesses: Although Random Forest is suitable for complex problems, it may not provide exact predictions for every instance. For example, the predicted values exhibit limited proximity to the actual results with a RMSE over 500. One reason for this is that the model is greatly influenced by the size of the dataset due to its nature. With limited data, the predictive performance decreases significantly.

Features of the Datasets: The dataset exhibits multicollinearity among the independent variables, which favored Random Forest due to its ability to handle correlated features effectively while the relatively small dataset size with fewer than 300 data points may have contributed to the variability in the results, particularly for Polynomial Regression, which can overfit when data is limited.

Decision Tree Regressor (Danica Zhang - Individual 3)

Strengths: The decision tree regressor is highly interpretable to explain easily the relationships between features and the target variable. Rather than linear regression, decision trees for regression can capture nonlinear relationships in the data. It can be relatively unaffected by the presence of outliers in the dataset.

Weaknesses: Decision trees can easily overfit the data, like capturing noise. CV MSE is 404105.33. Train MSE is 383046.45. Test MSE is 463032.63. CV MSE is larger than train MSE. It means that the model is overfitting and has a high variance. If the data changes a little, the model structure will be different. Thus, if the dataset size increases, the decision tree regression model in this report will not work. Decision tree regressor may not be suitable for complex relationships in the dataset. For complex relationships, other models may be used, such as poisson regression, polynomial regression, random forests, neural networks, and so on.

Limitations and Features Impacting Outcomes: The size of the dataset can significantly affect the outcomes. A small dataset can lead to overfitting, while a large dataset is more effective. Irrelevant or noisy independent variables can lead to poor outcomes. Although the decision tree regressor model can deal with missing data, however, the results may be changed. Hyperparameters and complexity of relationships also impacted the effectiveness of the decision tree regressor model.

Polynomial Regression (Emily and Jack - Individual 2 & 4)

Overview + strengths: Polynomial regression models extend from linear regression and allow for the modeling of more complex relationships between variables. It is able to capture non-linear patterns by introducing polynomial terms of various degrees, usually specified by the user for greatest model accuracy. This versatility and flexibility in controlling the model's fit makes polynomial regression valuable for more intricate data patterns that cannot be plotted by linear models. From the great spread of actual data values shown in the data visualization of our individual models, we can deduce that a simple linear model would be insufficient to capture the relationship to a great degree, thus making the choice of a polynomial regression more favorable.

Weaknesses: Polynomial regression, particularly at higher degrees, can exhibit overfitting issues, resulting in erratic predictions and reduced generalization performance when applied to new data, as illustrated in our evaluation where predictions on the scatter plot demonstrated decreased accuracy compared to the Random Forest regression model. Additionally, it grapples with multicollinearity and displays high sensitivity to data outliers if not removed. The model's complexity escalates with the degree chosen, rendering it less practical for real-world applications, displayed in the large MSE (245796 and 450252 for individual 1 and 2 respectively) and R2 scores (0.35 and 0.20) when the model is assessed against the training and testing data. Effective generalization often demands a sizable dataset, while in scenarios with limited data values (like our dataset with less than 500 values), employing a higher-degree polynomial model may not yield optimal results, as minor variations can lead to diverse interpretations.

Discussion:

In our report, we employed three machine learning methods—polynomial regression, decision trees, and random forests—to predict crime rates' relationship with socioeconomic factors. While polynomial regression allows for capturing non-linear relationships, its performance is limited due to potential concerns of overfitting. Decision trees, on the other hand, struggled with the continuous data's nuances. Therefore, random forests are considered as the best model for addressing our research question, excelling in capturing intricate correlations and resulting in the highest R-squared values.