Arrests Data Analysis in Burlington, VT: DS 2870 Final Project

Anna Rees

2024-06-19

```
knitr::opts_chunk$set(echo = TRUE,
                      warning = F,
                      message = F,
                      fig.align = "center")

# Load in the libraries to be used
pacman::p_load(tidyverse,
               skimr,
               ggtext,
               viridis,
               class,
               skimr,
               caret,
               rpart,
               rpart.plot,
               regclass,
               broom,
               GGally)
```

I. Introduction

Data Description:

The Arrests.csv file is a file of about 26,000 (25,917) arrests made in Burlington, Vermont from Jan 1, 2012 to January 8, 2024.

This dataset was created by the Burlington Vermont police department, and is sourced from the Burlington Data hub. This is an observational study of all arrests made in the 12 year scope of this dataset, so there is no sampling done. The data was created by the police reports given, so all information is detailed from the perspective of the reporting officer. As a resident of Burlington, I find it interesting to understand how local law enforcement and crime has changed over time. This dataset is uniquely interesting because it allows us to explore trends in law enforcement and crime over time in a specific city. Systematic racism is an undeniable aspect of our society, and this is most evident in Law enforcement and arrests across the United States – with Vermont being no exception. For this reason, I am omitting demographic information (race, ethnicity, and gender) from the dataset to avoid bias from the results for the purpose of this analysis.

The variables provided in the dataset are:

1. **incident_number:** STR The unique string identifier for the incident
2. **arrest_date:** Date of the event in “YEAR-MM-DD HR:MN:SC” format.
3. **gender:** Gender of the offender

4. **race:** Race of the offender
5. **age:** Age at the time of arrest
6. **charge:** Name and code of the charge
7. **most_serious:** [UNCLEAR]
8. **ethnicity:** Offender’s ethnicity
9. **felony:** Boolean of whether or not the charge is a felony
10. **violent:** Boolean of whether or not the crime was violent
11. **category:** Type of crime committed
12. **arrest_type:** Type of arrest
13. **ObjectId:** The unique Integer identifier for the incident

mutated/ added features:

1. **Date:** Date of the arrest in YYYY-MM-DD format
2. **Time:** Time of the arrest in HH:MM:SS format
3. **Hour:** Hour of the arrest, as an integer (0-23)

In order to appropriately assess this data, it must be loaded and cleaned. I removed the feature “most serious” as it was unclear what the purpose of this feature was, and separated out the arrest_date feature into date and time. I then selected only the features I felt were important to the analysis: ObjectId, Date, Time, category, violent, felony, charge, age, gender. I then filtered out any nan time or charge rows, since these are the essential variables in question.

Loading and Cleaning the data

```r
# Read the data and perform the necessary transformations
arrests <- read.csv("Arrests.csv", stringsAsFactors = TRUE) %>%
  # Separate arrest_date into Date, year, and Time features
  mutate(Date = as.Date(arrest_date, format = "%Y-%m-%d %H:%M:%S"),
         Time = format(as.POSIXct(arrest_date, format = "%Y-%m-%d %H:%M:%S"), "%H:%M:%S"),
         year = as.integer(format(as.POSIXct(arrest_date, format = "%Y-%m-%d %H:%M:%S"), "%Y")),
         Hour = as.integer(format(as.POSIXct(Time, format="%H:%M:%S"), "%H"))) %>%
  # Select only the important features
  select(ObjectId, Date, Time, Hour, category, violent, felony, charge, age, year) %>%
  # Remove rows with missing values
  drop_na() %>%
  # set logic vars to factors
  mutate(felony = as.factor(felony),
         violent = as.factor(violent))


# Display the cleaned dataset
str(arrests)
```

```
## 'data.frame':    18959 obs. of  10 variables:
##  $ ObjectId: int  2 3 4 5 6 8 9 10 11 12 ...
##  $ Date    : Date, format: "2012-01-01" "2012-01-01" ...
##  $ Time    : chr  "04:01:00" "04:30:00" "12:45:00" "12:45:00" ...
##  $ Hour    : int  4 4 12 12 12 17 17 20 20 23 ...
##  $ category: Factor w/ 12 levels "Assault","Disorder",..: 2 5 8 8 8 5 4 4 4 11 ...
##  $ violent : Factor w/ 2 levels "FALSE","TRUE": 1 2 1 1 1 2 1 1 1 1 ...
##  $ felony  : Factor w/ 2 levels "FALSE","TRUE": 1 1 2 1 1 2 2 2 1 1 ...
##  $ charge  : Factor w/ 237 levels "Accessory After the Fact 13 VSA 5 90z",..: 41 49 132 180 2
16 47 60 60 92 226 ...
##  $ age     : int  26 54 26 26 26 31 31 25 25 19 ...
##  $ year    : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
```

II. Data Visualizations

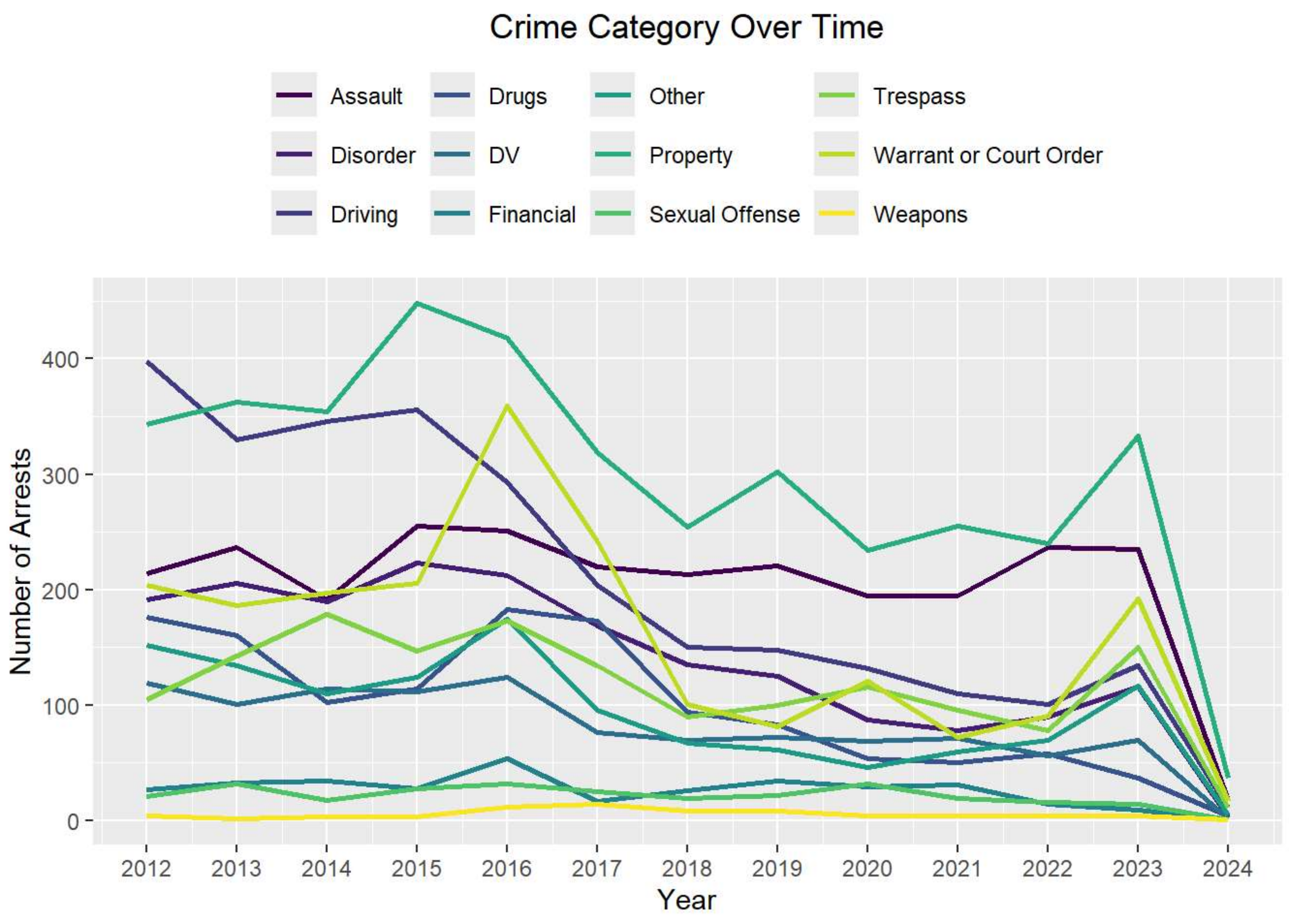
Write R code to create some relevant descriptive graphs, using techniques that we've used in class (ggplot, maybe dplyr). About 4 or 5 graphs should be plenty, depending on complexity.

For each graph and numerical summary, write a short description above the graph describing what the graph will be and what variables it will represent along with the purpose of the graph. Then below it, write a paragraph or two summarizing what you see, and suggesting some implications. For example, describe patterns that you observe in a graph, and suggest why they make sense, given what you know about the subject, or if they are unexpected. Do you think there is a cause-effect relationship between any variables? Explain your reasoning.

Crime category over time - line plot

This graph is exploring the number of arrests per year for different categories of crime

```
# aggregating data by yr and cat
crime_vs_age <- arrests %>%
  group_by(year, category) %>%
  summarise(count = n()) %>%
  ungroup()

crime_vs_age_plot <- ggplot(crime_vs_age,
                            aes(x = year, y = count, color = category)) +
  geom_line(size = 1) +
  scale_x_continuous(breaks = 2012:2024) +
  labs(title = "Crime Category Over Time",
       x = "Year",
       y = "Number of Arrests",
       color = "Crime Category") +
  scale_color_viridis_d() +
    theme(
      plot.title = element_text(hjust = 0.5), # Center the title
      legend.position = "top", # Place legend at the top
      legend.title = element_blank(), # Remove legend title
      legend.direction = "horizontal"
    )

crime_vs_age_plot
```

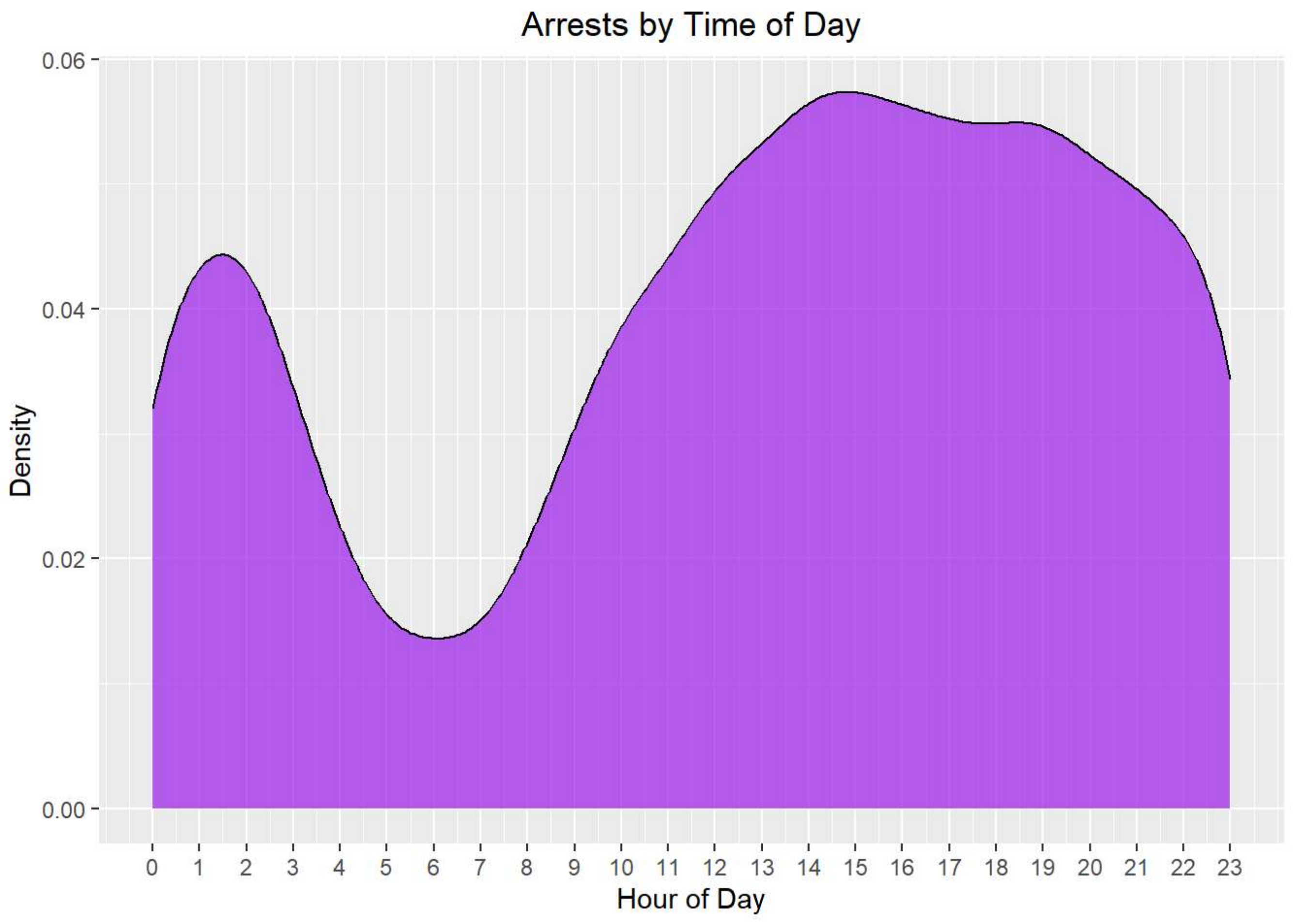


This line plot demonstrates the trend in types of crime over time. The trends in this plot may be interesting because it reflects the changes in crime being arrested for over time, so increases and decreases suggest changing and/or emerging social problems or law enforcement priorities in the city. The most common crime committed across all years is property related crime, but this trends down as we get closer to present day. Since 2024 data collection is ongoing, the counts for this year are obviously incomplete, and that is why there is a downturn in crime rates for this year specifically.

Arrests by time of day - density plot

I next will generate a density plot to show the distribution of arrests throughout the day.

```
arrests_by_TOD <- ggplot(arrests, aes(x = Hour)) +
  geom_density(fill = "purple", alpha = 0.7) +
  scale_x_continuous(breaks = 0:23) +
  labs(title = "Arrests by Time of Day",
       x = "Hour of Day",
       y = "Density") +
  theme(
    plot.title = element_text(hjust = 0.5)
  )

arrests_by_TOD
```

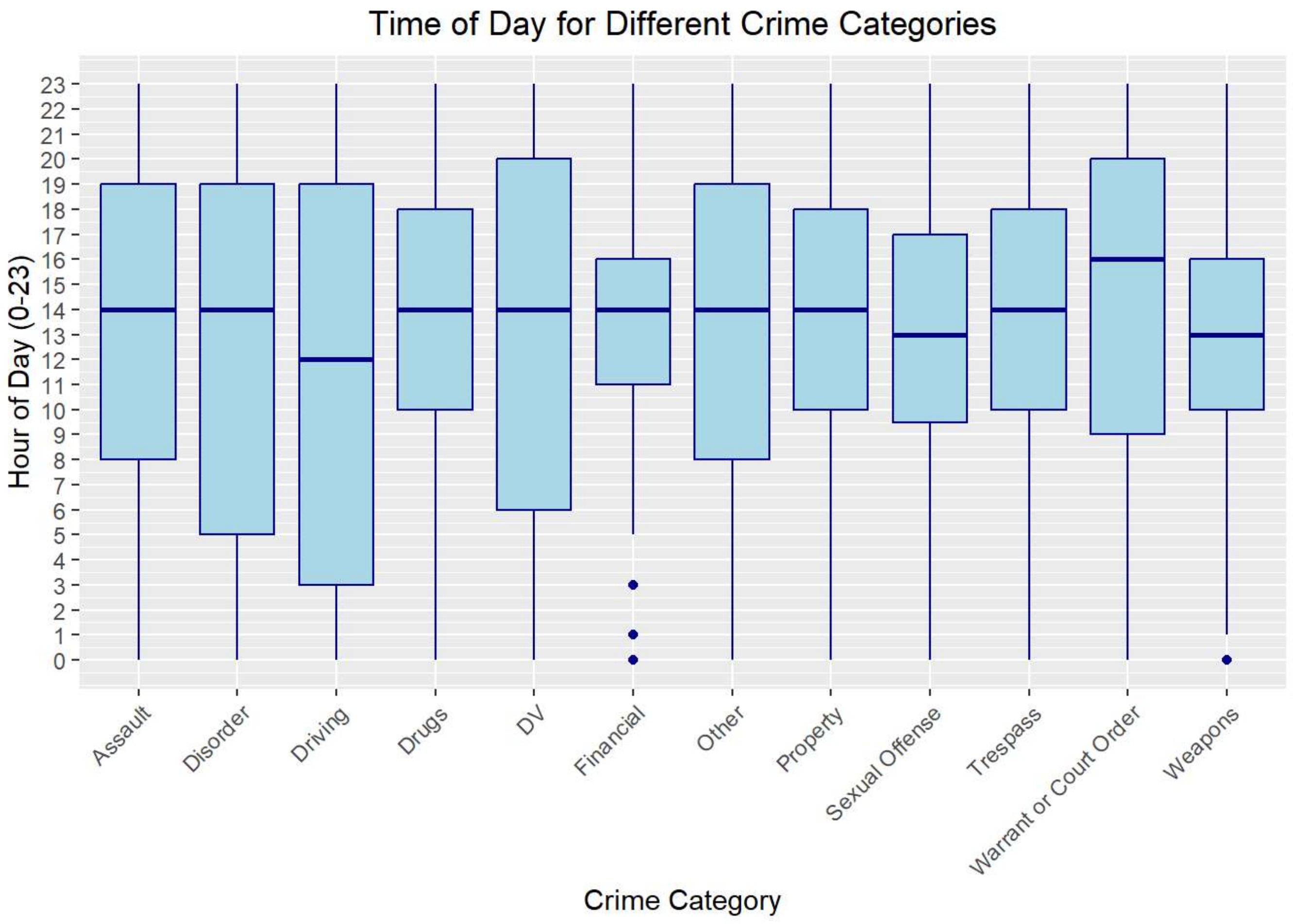


There appears to be a downturn in crime rates between the hours of 2-10 am, with the fewest arrests occuring at 6 in the morning. The most arrests occur in the afternoon, with a peak at 15 (3 pm).

When do each category of crime typically occur during the day? - Boxplot

I will create a boxplot depicting each category of crime and their quartile distribution of the time of day they typically occur, by the hour

```
box_plot <- ggplot(arrests, aes(x = category, y = Hour)) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  scale_y_continuous(breaks = 0:23) +  # Set y-axis to show each hour
  labs(title = "Time of Day for Different Crime Categories",
       x = "Crime Category",
       y = "Hour of Day (0-23)") +
  theme(axis.text.x = element_text(angle = 45,
                                   hjust = 1),
        plot.title = element_text(hjust = 0.5)
        )

box_plot
```

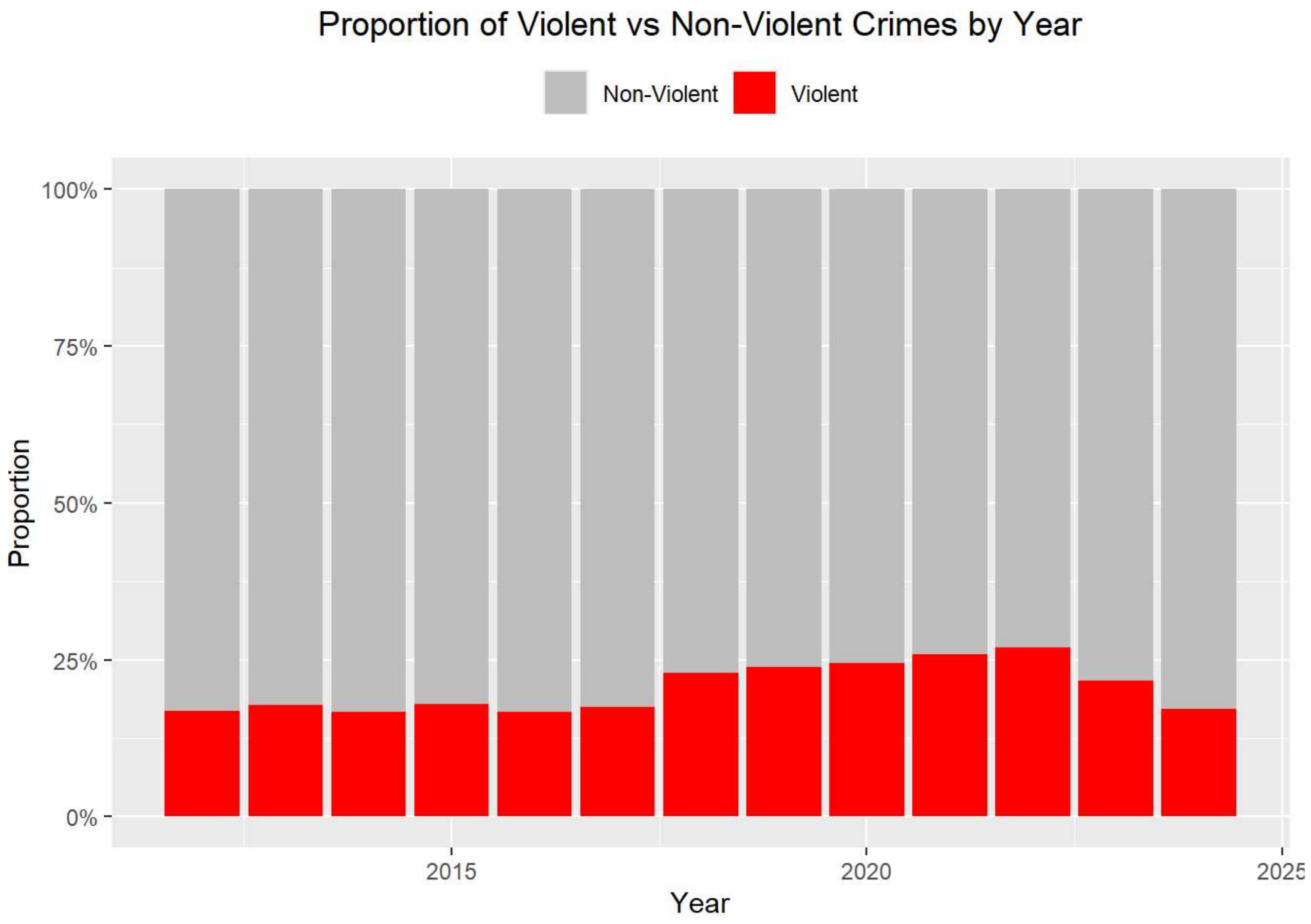


Based on this boxplot, there is little variation in the median hour of the day for arrests based on the crime type. That being said, driving related arrests tend to occur later than other kinds of arrests, but with a relatively wide spread in times. Warrant or court-order arrests tend to occur slightly earlier than other arrests, based on the median time.

Proportion of violent vs non-violent crimes - bar chart

I am next curious to determine if the proportion of violent vs nonviolent crime has changed over the years, using a stacked bar chart.

```
# Calculate proportions by year and violent status
violent_crime_proportion_year <- arrests %>%
  group_by(year, violent) %>%
  summarise(count = n()) %>%
  mutate(proportion = count / sum(count))

# creating the stacked bar plot
violence_prop_plot <- ggplot(violent_crime_proportion_year, aes(x = year, y = proportion, fill =
factor(violent, labels = c("Non-Violent", "Violent")))) +
  geom_bar(stat = "identity", position = "fill") +
  scale_fill_manual(values = c("grey", "red")) +
  labs(title = "Proportion of Violent vs Non-Violent Crimes by Year",
       x = "Year",
       y = "Proportion",
       fill = "Crime Type") +
  scale_y_continuous(labels = scales::percent_format()) +
  theme(
    plot.title = element_text(hjust = 0.5), # Center the title
    legend.position = "top", # Place legend at the top
    legend.title = element_blank(), # Remove legend title
    legend.direction = "horizontal"
  )

violence_prop_plot
```

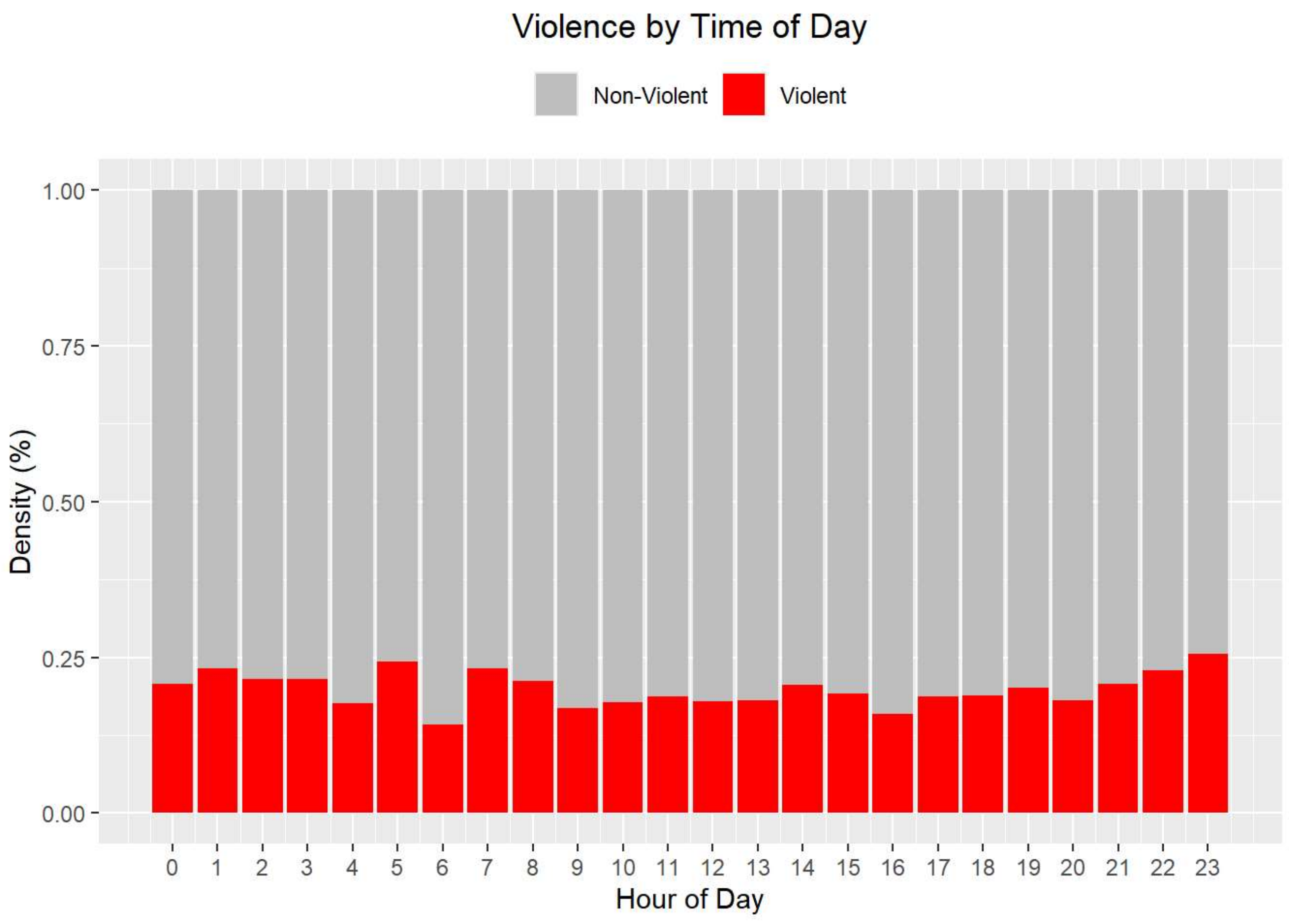


Based on this plot, there appears to be an uptick in violent crime between 2018 and 2022, with this upward trend finally breaking in 2023. It would be interesting in future work to investigate what societal changes are occurring during this time, especially with the COVID-19 pandemic in mind, that increased the proportion of violent crime during this time.

What time of day experiences the most violent crime? - bar plot

Does the time of day have any indication into the likelihood of a violent crime?

```r
# Calculate the density of arrests by hour for both violent and non-violent crimes
density_data <- arrests %>%
  group_by(Hour, violent) %>%
  summarise(count = n()) %>%
  group_by(Hour) %>%
  mutate(total_count = sum(count),
         density = count / total_count * 100)

# Plot the densities for both violent and non-violent crimes
violence_by_TOD <- ggplot(density_data, aes(x = Hour, y = density, fill = factor(violent))) +
  geom_bar(stat = "identity", position = "fill") +
  scale_fill_manual(values = c("grey", "red"), labels = c("Non-Violent", "Violent")) +
  labs(title = "Violence by Time of Day",
       x = "Hour of Day",
       y = "Density (%)") +
  scale_x_continuous(breaks = 0:23) +
  theme(
    plot.title = element_text(hjust = 0.5), # Center the title
    legend.position = "top", # Place legend at the top
    legend.title = element_blank(), # Remove legend title
    legend.direction = "horizontal"
  )

violence_by_TOD
```



The proportion of violent vs nonviolent crime does not appear to follow a trend based on the time of day. That being said, 6 am seems to have the lowest proportion violent crime compared to any other time. While there is slight variation in the proportions of violent crime throughout a given day, it is relatively consistent.

III. Machine Learning Methods

Multiple Linear Regression

Predicting the time of the offense by other factors

Using a linear model, I am interested in predicting the time of arrest based on the other factors. Based on the kinds of charge, the year it occured, and other features in the data, is it possible to accurately predict the time of day of the arrest?

First, I want to train the lm based on different possible features, and determine which model is the most accurate. I will fit several linear models with the following names and explanatory variables:

1. ‘hour_lm1’: category + violent + felony + age + Date + year
2. ‘hour_lm2’: category + violent + felony + age
3. ‘hour_lm3’: category + violent + felony
4. ‘hour_lm4’: category
5. ‘hour_lm5’: category + age + Date + year
6. ‘hour_lm6’: age + Date + year
7. ‘hour_lm7’: age + violent

Note, I have omitted “charge” as a feature. Since there are so many levels to the factor (237 different charges in the dataset), the train and test data is unable to split where all charges are represented.

I will first split the dataset into training and testing groups

```r
set.seed(123)
# Split data into training and testing sets
train_index <- createDataPartition(arrests$Hour, p = 0.7, list = FALSE)
train_data <- arrests[train_index, ]
test_data <- arrests[-train_index, ]


# Fit the linear models based on different explanatory variables
hour_lm1 <- lm(Hour ~ category + violent + felony + age + Date + year, data = train_data) # most
features
hour_lm2 <- lm(Hour ~ category + violent + felony + age, data = train_data) # category, violenc
e, age
hour_lm3 <- lm(Hour ~ category + violent + felony, data = train_data) # category, violence
hour_lm4 <- lm(Hour ~ category, data = train_data) # only age, hour, year
hour_lm5 <- lm(Hour ~ category + age + Date + year, data = train_data)
hour_lm6 <- lm(Hour ~ age + Date + year, data = train_data)
hour_lm7 <- lm(Hour ~ age + violent, data = train_data)

# Assessing the models
model_assessment <- bind_rows(
  .id = "model",
  "hour_lm1" = glance(hour_lm1),
  "hour_lm2" = glance(hour_lm2),
  "hour_lm3" = glance(hour_lm3),
  "hour_lm4" = glance(hour_lm4),
  "hour_lm5" = glance(hour_lm5),
  "hour_lm6" = glance(hour_lm6),
  "hour_lm7" = glance(hour_lm7)
) %>%
  dplyr::select(model, n_predictors = df, r.squared, sigma) %>%
  mutate(r.squared = round(r.squared, 4),
         sigma = round(sigma, 0)) %>%
  gt::gt()

model_assessment
```

model	n_predictors	r.squared	sigma
hour_lm1	16	0.0152	7
hour_lm2	14	0.0144	7
hour_lm3	13	0.0137	7
hour_lm4	11	0.0133	7
hour_lm5	14	0.0147	7
hour_lm6	3	0.0019	7

model	n_predictors	r.squared	sigma
hour_lm7	2	0.0007	7

All of the models had the same sigma, but the 1st, 2nd, and 5th models created had the highest R^2 at 0.0152, 0.0144, and 0.0147 respectively. When the r squared value is closer to 1, this indicates a better fit so the model explains a large portion of the variance in the given variables.

```
hour_preds <- tibble(
  hour = test_data$Hour,
  hour1 = predict(object = hour_lm1, newdata = test_data),
  hour2 = predict(object = hour_lm2, newdata = test_data),
  hour5 = predict(object = hour_lm5, newdata = test_data)
)

tibble(hour_preds)
```

```
## # A tibble: 5,686 × 4
##     hour hour1 hour2 hour5
##    <int> <dbl> <dbl> <dbl>
##  1    16  12.7  12.9  12.6
##  2    16  13.3  13.5  13.3
##  3    23  12.4  12.6  12.7
##  4    18  12.8  12.9  12.7
##  5     1  11.1  11.2  11.1
##  6     5  12.9  13.0  12.7
##  7     8  12.7  12.9  12.6
##  8    21  13.7  13.9  13.6
##  9    22  13.4  13.5  13.4
## 10    15  13.5  13.7  13.5
## # ℹ 5,676 more rows
```

Calculating the R2 and MAE for the test data:

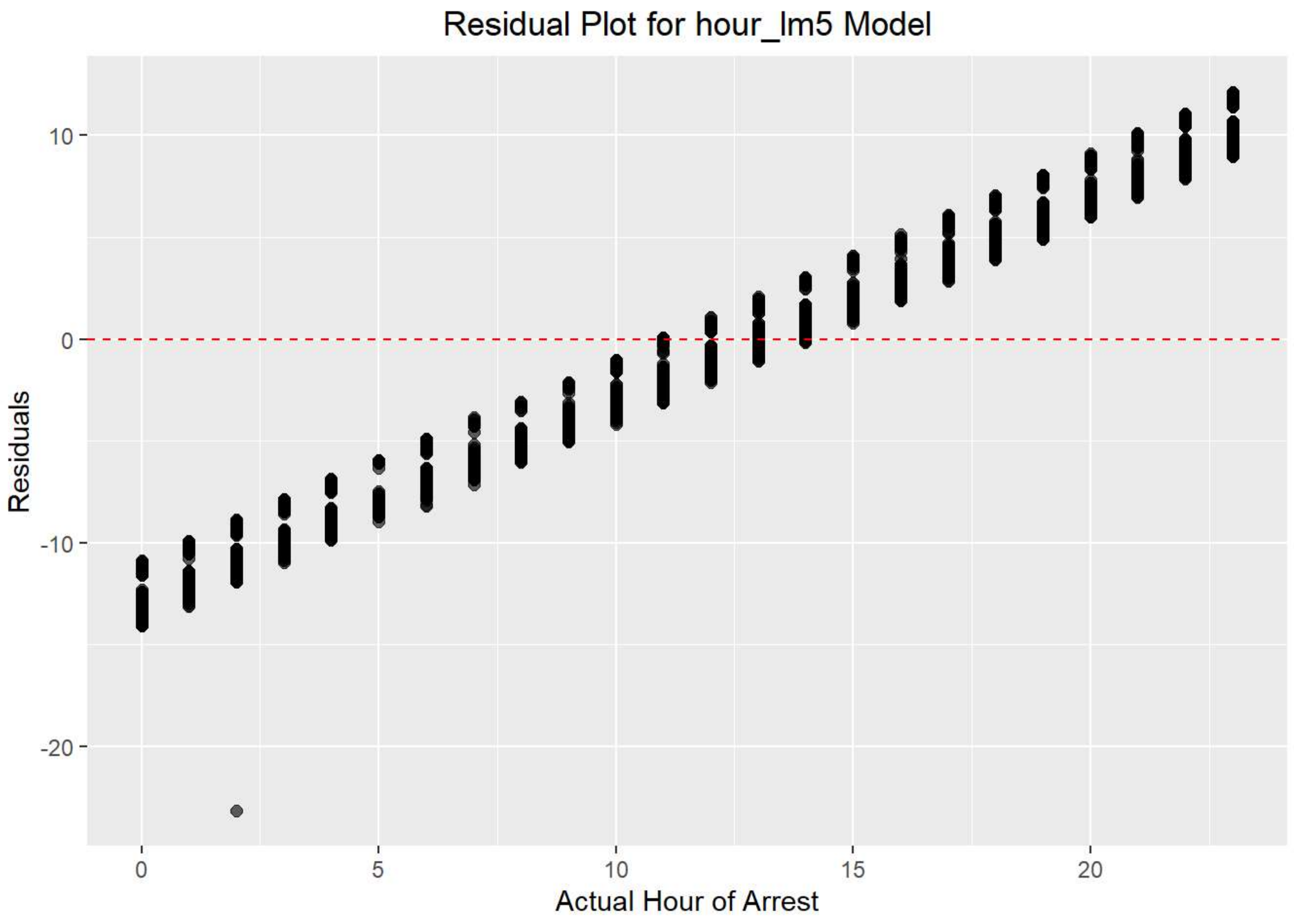
```
Model = c("hour1", "hour2", "hour5")

# Calculate R-squared for each model
r_squared <- tibble(
  Model = Model,
  R_squared = c(
    cor(test_data$Hour, hour_preds$hour1)^2,
    cor(test_data$Hour, hour_preds$hour2)^2,
    cor(test_data$Hour, hour_preds$hour5)^2
  )
)

# Calculate sigma for each model
sigma <- tibble(
  Model = Model,
  Sigma = c(
    sqrt(sum((test_data$Hour - hour_preds$hour1)^2) / nrow(test_data)),
    sqrt(sum((test_data$Hour - hour_preds$hour2)^2) / nrow(test_data)),
    sqrt(sum((test_data$Hour - hour_preds$hour5)^2) / nrow(test_data))
  )
)

# Calculate MAE for each model
mae <- tibble(
  Model = Model,
  MAE = c(
    sum(abs(test_data$Hour - hour_preds$hour1)) / nrow(test_data),
    sum(abs(test_data$Hour - hour_preds$hour2)) / nrow(test_data),
    sum(abs(test_data$Hour - hour_preds$hour5)) / nrow(test_data)
  )
)

# Combine the results into a single data frame
test_results <- bind_rows(r_squared, sigma, mae)

# Display the results
test_results
```

```
## # A tibble: 9 × 4
##   Model R_squared Sigma   MAE
##   <chr>     <dbl> <dbl> <dbl>
## 1 hour1    0.0115 NA    NA
## 2 hour2    0.0105 NA    NA
## 3 hour5    0.0116 NA    NA
## 4 hour1   NA       6.84 NA
## 5 hour2   NA       6.85 NA
## 6 hour5   NA       6.84 NA
## 7 hour1   NA      NA     5.71
## 8 hour2   NA      NA     5.71
## 9 hour5   NA      NA     5.71
```

- lm5 has the highest R2 at 0.0116 which indicates best fit
- lm5 has the lowest sigma at 6.8410 which indicates better accuracy
- lm2 has the lowest MAE value, which indicates best accuracy

Since lm5 performed the best in 2/3 metrics, I will pursue lm5 for the residual plot.

```
# Predictions on test data using hour_lm5
hour_preds <- predict(hour_lm5, newdata = test_data)

# Calculate residuals
residuals <- test_data$Hour - hour_preds

# Create a data frame for plotting
residual_plot_data <- tibble(
  Hour = test_data$Hour,
  Residuals = residuals
)

# Create the residual plot
residual_plot <- ggplot(residual_plot_data, aes(x = Hour, y = Residuals)) +
  geom_point(alpha = 0.6, size = 2) +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(
    x = "Actual Hour of Arrest",
    y = "Residuals",
    title = "Residual Plot for hour_lm5 Model"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5)
  )

residual_plot
```



Based on the residuals, this linear model does **not** accurately predict hour of arrest. There is a very clear strong positive linear relationship between the residuals and the hour of arrest, which violates the linear assumption. There is one outliter near the 2 hour of arrest, which suggests this observation had a different residual than expected.

Regression trees

I want to be able to predict the time of day (hour) an arrest occurred using the most important features, as determined by the linear models made above: category + age + Date + year

Fitting the full tree

```
arrests_reg <- arrests %>%
  dplyr::select(Hour, category, age, Date, year)

# splitting into test and train
train_index <- createDataPartition(arrests_reg$Hour, p = 0.7, list = FALSE)
train_data <- arrests_reg[train_index, ]
test_data <- arrests_reg[-train_index, ]

arrest_tree_full <-
  rpart(
    formula = Hour ~ .,
    data = train_data,
    method = "anova",
    minsplit = 2,
    minbucket = 1,
    cp = -1
  )

arrest_tree_full$cptable %>%
  data.frame() %>%
  tail(n = 10)
```

```
##                 CP nsplit  rel.error   xerror       xstd
## 2706  2.662025e-07  10747 0.01856423 1.632690 0.02121878
## 2707  2.662025e-07  10748 0.01856396 1.632690 0.02121878
## 2708  2.662025e-07  10752 0.01856290 1.632690 0.02121878
## 2709  2.662025e-07  10755 0.01856210 1.632690 0.02121878
## 2710  2.662025e-07  10757 0.01856157 1.632690 0.02121878
## 2711  2.662025e-07  10759 0.01856104 1.632690 0.02121878
## 2712  2.129620e-07  10761 0.01856050 1.632690 0.02121878
## 2713  1.331012e-07  10762 0.01856029 1.632707 0.02121879
## 2714  1.331012e-07  10763 0.01856016 1.632695 0.02121864
## 2715 -1.000000e+00  10764 0.01856003 1.632695 0.02121864
```

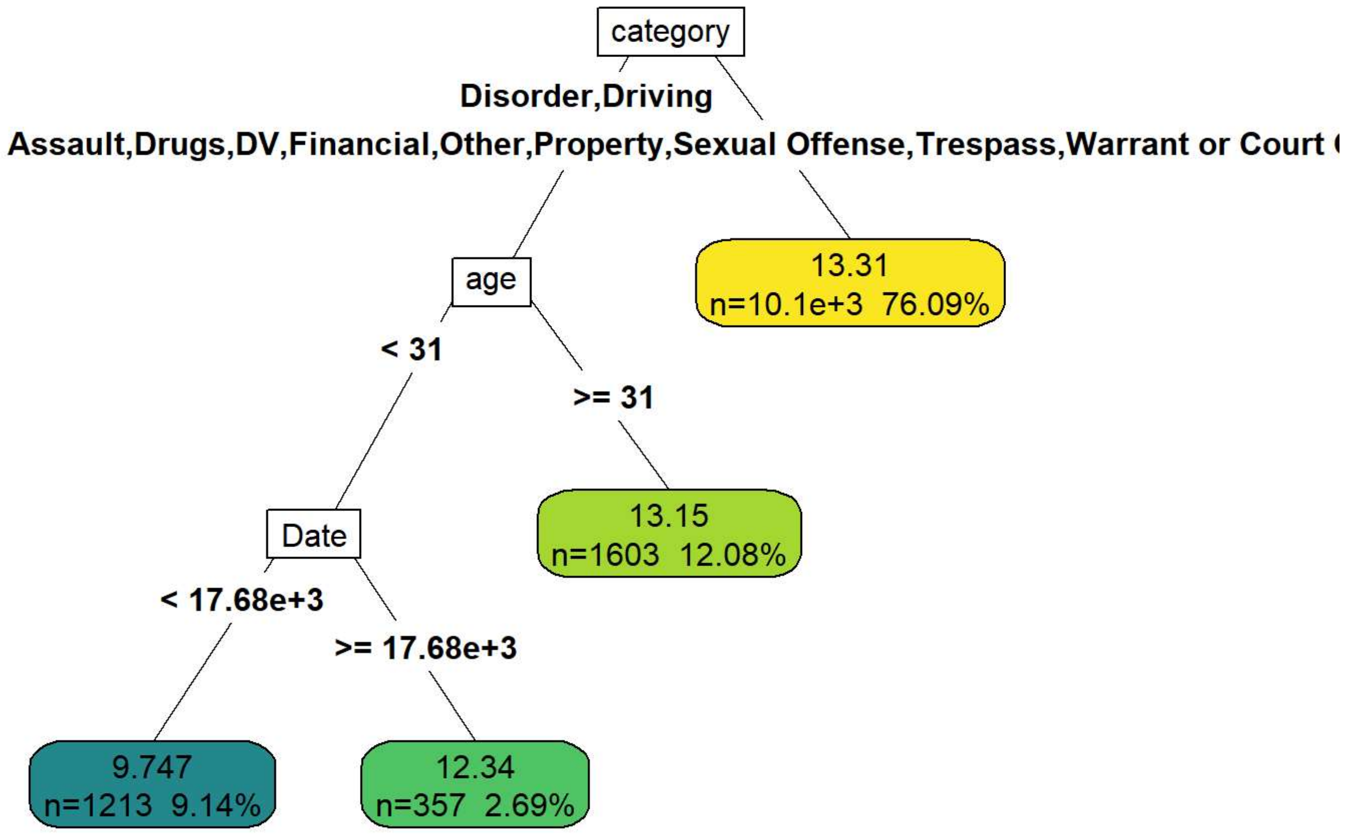
Finding the pruning Point

```
# finding xerror cutoff: min(xerror) + xstd
arrest_tree_full$cptable %>%
  data.frame() %>%
  #find row w smallest xerror
  slice_min(xerror, n = 1, with_ties = F) %>%
  # using muatate to create xerror + xs
  mutate(xerror_cutoff = xerror + xstd) %>%
  # using pull to get xerror cutoff as a single numeric val
  pull(xerror_cutoff) ->
  xcutoff

# Now we find the cp value to prune the tree:
arrest_tree_full$cptable %>%
  data.frame() %>%
  # use fitler to pick rows less than xcutoff val
  filter(xerror < xcutoff) %>%
  # use slice to keep the fitrst row
  slice(1) %>%
  # SAVE CP VAL
  pull(CP) ->
  prune_cp

c("The CP value is:", round(prune_cp, 4))
```

```
## [1] "The CP value is:" "0.0022"
```

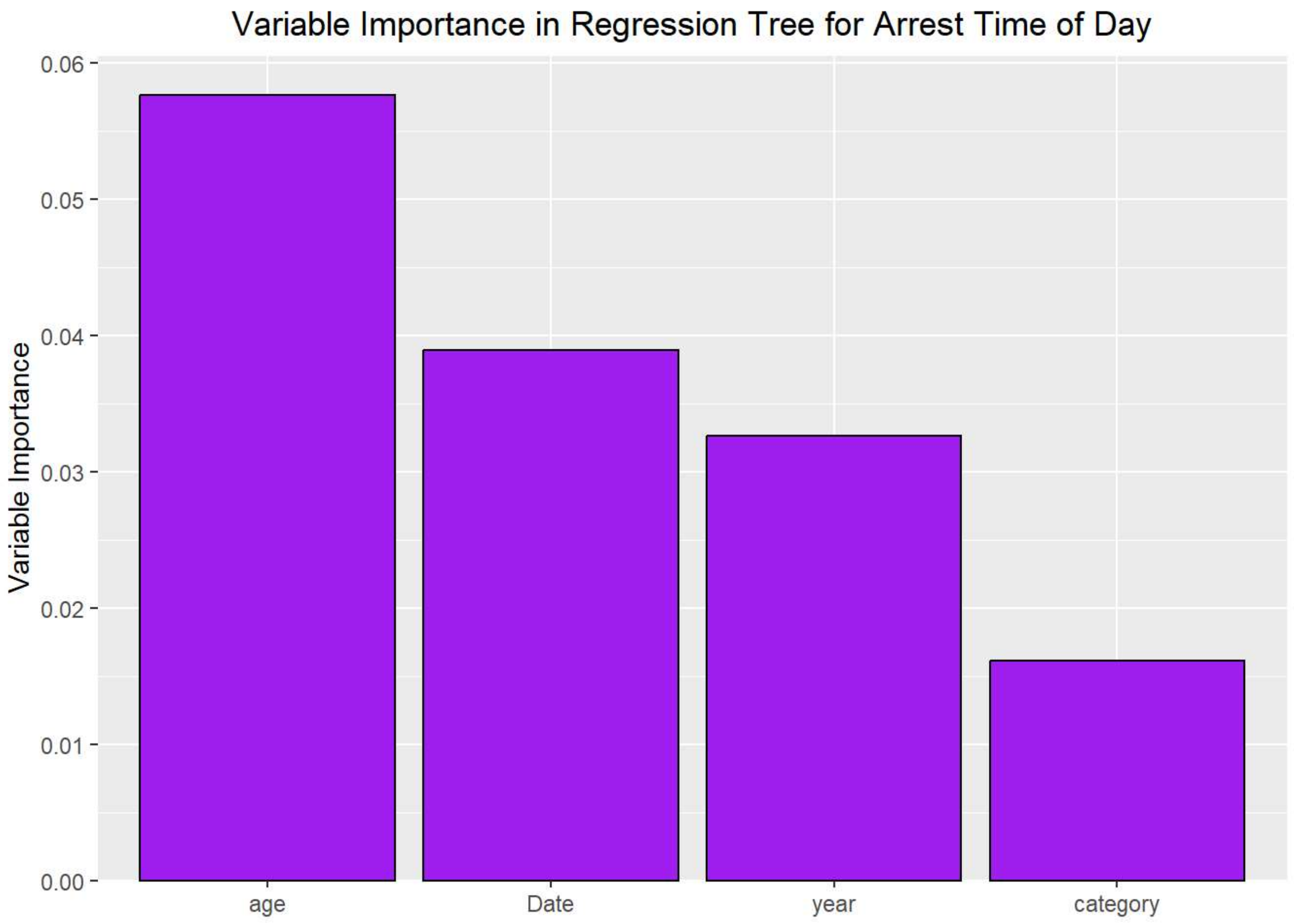
Pruning and Plotting

```
# pruning...
prune(tree = arrest_tree_full,
      cp = prune_cp) ->
  arrest_tree

# plotting...
rpart.plot(arrest_tree,
           digits = 4,
           fallen.leaves = FALSE,
           type = 5,
           extra = 101,
           box.palette = 'BlGnYl')
```



Variable Importance

```
caret::varImp(arrest_tree) |>
  arrange(desc(Overall)) |>
  rownames_to_column(var = "variable") |>

  ggplot(mapping = aes(x = fct_reorder(variable, -Overall),
                       y = Overall)) +

  geom_col(fill = "purple",
           color = "black") +

  labs(title = "Variable Importance in Regression Tree for Arrest Time of Day",
       x = NULL,
       y = "Variable Importance") +

  scale_y_continuous(expand = c(0, 0, 0.05, 0)) +
  theme(
    plot.title = element_text(hjust = 0.5)
  )
```



When predicting the time of day an arrest occured, the 3 most important features are the age of the offender, the date of the offense, and the year the offense occurred.

Predicting hour on test data

I want to use both the pruned and full trees to predict the hour of arrest for the testing set

```
# full tree
predict(
  object = arrest_tree_full,
  newdata = test_data
) ->
  full_tree_pred

# pruned tree
predict(
  object = arrest_tree,
  newdata = test_data
) ->
  pruned_tree_pred

# Combine the predictions with the actual prices in the test dataset
test_results <- test_data %>%
  mutate(full_tree_pred = full_tree_pred,
         pruned_tree_pred = pruned_tree_pred,
         actual_hour = test_data$Hour)

# Calculate R^2 for the full tree
R2_full_tree <- cor(test_results$full_tree_pred, test_results$actual_hour)^2

# Calculate R^2 for the pruned tree
R2_pruned_tree <- cor(test_results$pruned_tree_pred, test_results$actual_hour)^2

# Output the R^2 values
c("Full tree R2:", round(R2_full_tree, 4))
```

```
## [1] "Full tree R2:" "0.0327"
```

```
c("Pruned tree R2:", round(R2_pruned_tree, 4))
```

```
## [1] "Pruned tree R2:" "0.0208"
```

Surprisingly, the full tree performed better than the pruned tree. This may be because the full tree has a higher training accuracy, which was an asset to minimize training error and capturing all patterns and relationships in the data. The pruned tree may have lost some of that detail that outweigh any benefits it provided to over-fitting.

IV. Conclusions

The goal of my project was the evaluate the arrest data for the city of Burlington, VT to gain insights into patterns of arrest, especially focusing on the time of day arrests occur. A linear regression model proved to be unhelpful in gaining insight into this, but generating the linear models enabled me to determine what features were most valuable to this model, and I took these and used them for a regression tree. This regression tree was more enlightening, and helped to determine that age was the most important factor when trying to determine the time of day an arrest may occur

V. Limitations / Recommendations

My study had several inherent limitations. Firstly, the data was limited to arrests in Burlington, BY, which may not be representative of broader trends in arrests for other regions. Additionally, the data contained many categorical variables with varied levels such as the charge feature, which created some complexity and limitations in the train/test split.

In the future, expanding the dataset to multiple cities or regions could help gain insight into more generalizeable arrest trends. Like what was mentioned before, it would be interesting in future work to investigate what societal changes are occurring during the time of this study, especially with the COVID-19 pandemic in mind, that may have influenced the arrests in the area.