Exploratory Data Analysis

- StudentID: 22000724

- Name: Eunbi Cho

- 1st Major: Management

- 2nd Major: AI Convergence

The proposed project is to analyze Dallas' crime dataset using exploratory data analysis (EDA) techniques using Python programming. The project involves performing a variety of EDA tasks on the dataset, including data cleaning, univariate analysis, and multivariate analysis. The goal of the project is to propose the potential project idea based on the the insights obtained from the patterns and trends in the crime data.

1. Data Explanation

Data explanation consists of descriptives statistics on overall data, including sample size, number of variables, data type, data range, distribution, etc.

Below is the representation of Dallas crime raw data, that is used for this EDA.

```
#Load the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#Load the data
df = pd.read_stata("raw_Dallas.dta")

#View the data
df.head()
```

incidentnum	UCR_ctype	cnt	ctype	year	servicenumberid	watch	call911problem	typeofincident	typelocation	...
000001-2019	NaN	1		2019	000001-2019-01	3	ODJ - OFF DUTY JOB	BMV	Parking Lot (All Others)	...
000001-2020	NaN	1		2020	000001-2020-01	1	58 - ROUTINE INVESTIGATION	PUBLIC INTOXICATION	Highway, Street, Alley ETC	...
000002-2015	1.0	1	MURDER	2015	000002-2015-01	3	19 - SHOOTING	MURDER	Other	...
000002-2018	NaN	1	FOUND	2018	000002-2018-01	1	58 - ROUTINE INVESTIGATION	RECOVERED OUT OF TOWN STOLEN VEHICLE (NO OFFENSE)	Highway, Street, Alley ETC	...
000002-2019	NaN	1		2019	000002-2019-01	1	19 - SHOOTING	INJURED PERSON - FIREARM INJURY (NO OFFENSE)	Single Family Residence - Occupied	...

Data is ready to be explored

```
df.shape
#(663249, 107)
```

The Dallas crime data consists of 107 variable columns in 663249 rows.

```
df.info()
'''
Int64Index: 663249 entries, 0 to 663248
Columns: 107 entries, incidentnum to blkidfp10
dtypes: float64(1), int16(1), int8(1), object(104)
memory usage: 538.3+ MB
'''
```

The `info` function provides the basic information about the dataset:

Out of 107 columns, all data types in a raw form is object, except “UCR_ctype” in float 64, “cnt” in int8 and “year” in int16.

```
print(df.isnull().sum())
print(df.isnull().sum().sum())
```

By using `isnull` function, we checked that there are only 420075 values missing in the “UCR_ctype” column out of whole dataset. However by comparing with the data summary from head function, other empty data were found.

```
df.describe(include='all')
```

	incidentnum	UCR_ctype	cnt	ctype	year	servicenumberid	watch	call911problem	typeofin
count	663249	243174.000000	663249.0	663249	663249.000000	663249	663249	663249	663249
unique	663249	NaN	NaN	54	NaN	663249	3	118	1055
top	000001-2019	NaN	NaN		NaN	000001-2019-01	1	58 - ROUTINE INVESTIGATION	BMV
freq	1	NaN	NaN	297247	NaN	1	263685	75867	75861
mean	NaN	4.641907	1.0	NaN	2017.409422	NaN	NaN	NaN	NaN
std	NaN	1.048904	0.0	NaN	1.917947	NaN	NaN	NaN	NaN
min	NaN	1.000000	1.0	NaN	2014.000000	NaN	NaN	NaN	NaN
25%	NaN	4.000000	1.0	NaN	2016.000000	NaN	NaN	NaN	NaN
50%	NaN	5.000000	1.0	NaN	2018.000000	NaN	NaN	NaN	NaN
75%	NaN	5.000000	1.0	NaN	2019.000000	NaN	NaN	NaN	NaN
max	NaN	6.000000	1.0	NaN	2020.000000	NaN	NaN	NaN	NaN

Function `describe` was used to provide descriptive statistics for the numerical variables listed above. Moreover, by using `(include='all')` , the number of unique values, the first value and the frequency of variables in objec type were given.

To take a look at a distribution of crimes over the time, following line plot was graphed:

```
df["date1ofoccurrence"] = pd.to_datetime(df["date1ofoccurrence"])
df["date2ofoccurrence"] = pd.to_datetime(df["date2ofoccurrence"])
df["dateofreport"] = pd.to_datetime(df["dateofreport"])

df["year"] = df["date1ofoccurrence"].dt.strftime("%Y")

incidents_by_year = df.groupby("year")['incidentnum'].count()

plt.plot(incidents_by_year.index, incidents_by_year.values)
plt.xlabel("Year")
plt.ylabel("Number of Incidents")
plt.xticks(rotation=45)
plt.show()
```

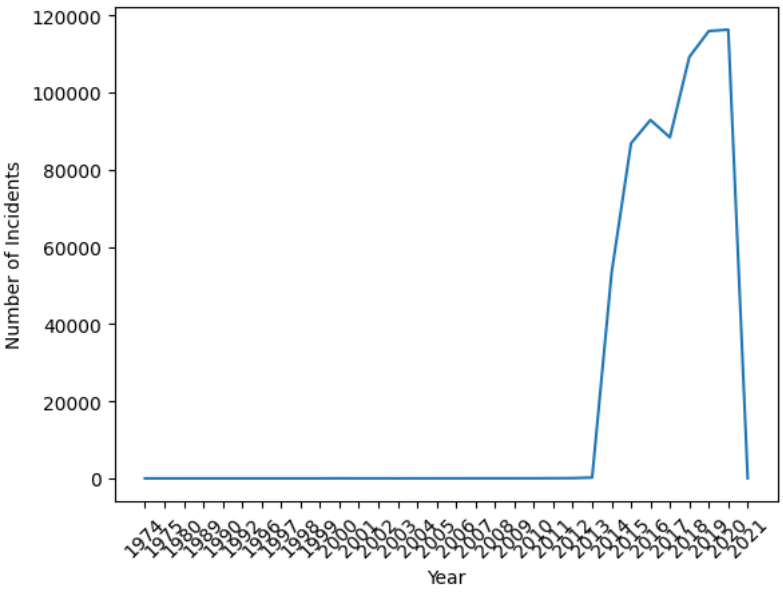

Figure 1. Crime Distribution

The following graph, indicates that the data is in between the range of 1974~ 2021, having most of its number in between 2014~2020.

2. Univariate Analysis

When analyzing a dataset, it's important to identify the key variables that are relevant to the analysis. While there may be many variables in a dataset, it's important to focus on the variables, which will help to ensure that the analysis is more focused and efficient.

In this case, an univariate analysis was performed to understand the key variables from different aspects.

2.1 Type of Incident

```
plt.figure(figsize=(20,6))

# Get top 10 types of incidents in descending order
top_10_incidents = df['typeofincident'].value_counts().sort_values(ascending=False).head(10).index.tolist()

# Filter dataframe to only include top 10 incidents
df_top_10 = df[df['typeofincident'].isin(top_10_incidents)]

# Plot countplot with filtered dataframe
sns.countplot(x='typeofincident', data=df_top_10, order=top_10_incidents)

plt.xticks(rotation=90)
plt.title('Distribution of Top 10 Crime Types')
plt.show()
```

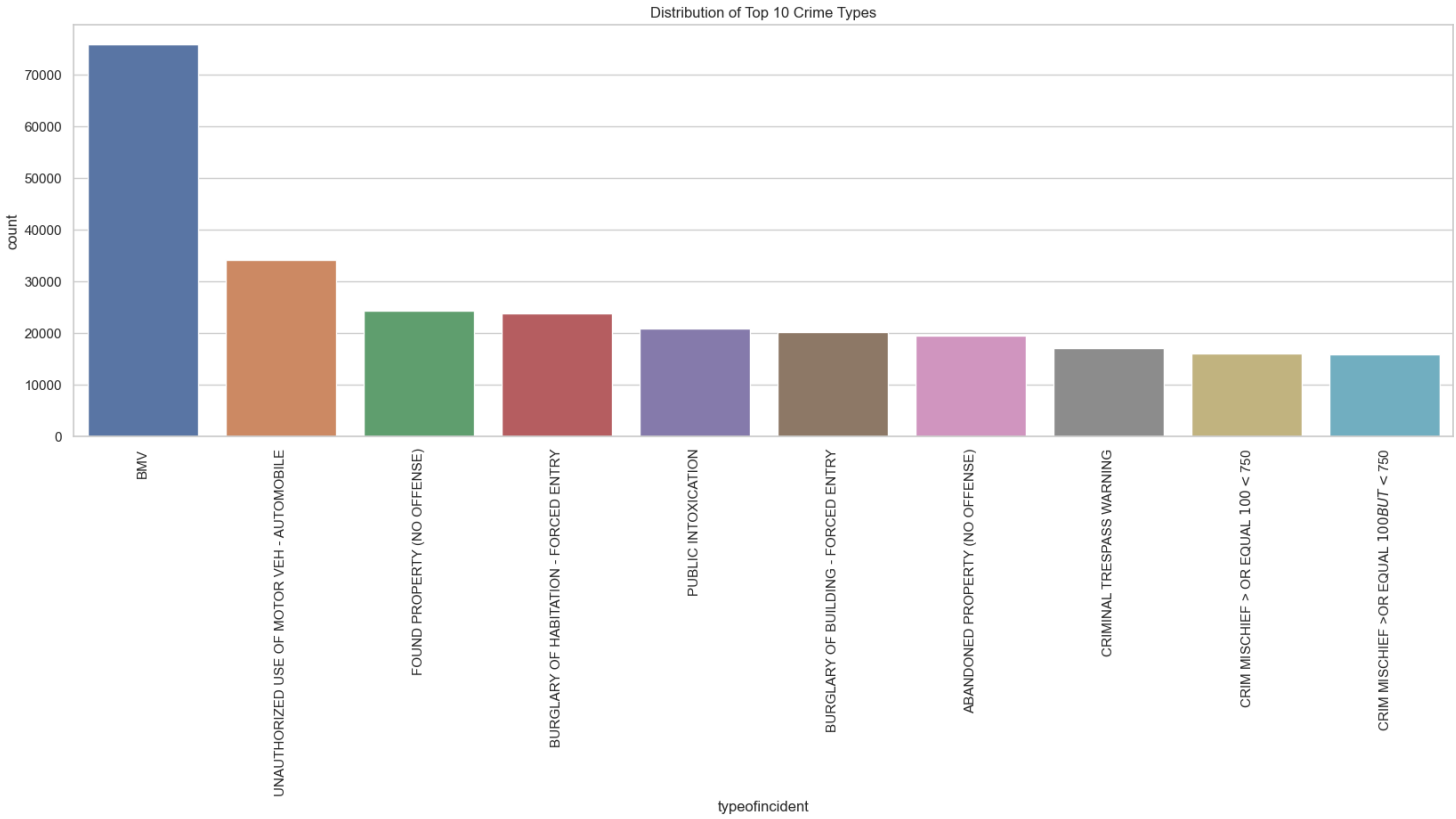

Figure 2. Distribution of Top 10 Crime Types

From analysing the distribution of Top 10 crime types of Dallas, it was found that BMV, which stands for Burglary of a Motor Vehicle, was the most occuring crime.

However, to gain a deeper understanding of the crime issue in Dallas, we studied several additional variables.

2.2 Day of Week

```
import matplotlib.cm as cm

# Create a new dataframe with columns for day of week and count of crimes
df_day = df['day1oftheweek'].value_counts().sort_index().reset_index()
df_day.columns = ['day1oftheweek', 'count']

# Define the order of days of the week
order = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']

# Define the size of the plot
plt.figure(figsize=(10, 6))

# Create a colormap
my_cmap = cm.get_cmap('Blues')

# Set the range of values for the colormap
vmin = df_day['count'].min()
vmax = df_day['count'].max()

# Normalize the count values to the range [0, 1]
norm = plt.Normalize(vmin=vmin, vmax=vmax)

# Create a list of colors for the bars based on the count values
colors = [my_cmap(norm(value)) for value in df_day['count']]

# Create the bar plot
plt.bar(df_day['day1oftheweek'], df_day['count'], color=colors, align='center')

# Set the order of days of the week
plt.xticks(df_day['day1oftheweek'], order)

# Set the title and axis labels
plt.title('Crime Count by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Count')

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

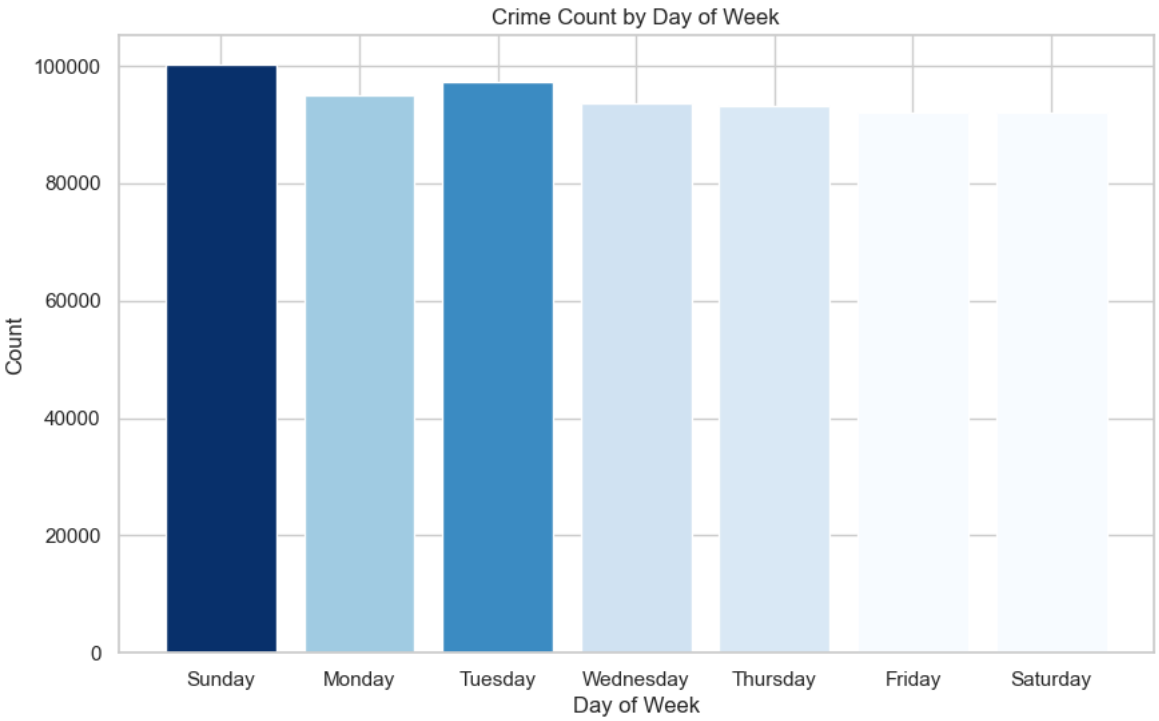

Figure 3. Crime Count by Day of Week

After analyzing the crime data in Dallas, it was found that most crimes occur on Sundays. This finding was surprising because it was initially expected that crime rates would be highest on weekends starting from Fridays.

To fully understand the crime patterns in Dallas, it is essential to analyze the data from different perspectives and compare the findings in relation to other variables, such as the type of crime and the time of day. This multivariate approach can provide a more comprehensive understanding of the underlying causes of crime and help develop effective strategies to prevent and reduce crime in the city.

3. Multivariate Analysis

Multivariate analysis refers to the statistical analysis of data that involves multiple variables or factors. In other words, it is the analysis of data that involves examining the relationships between more than two variables simultaneously.

By analysing hidden patterns between variables, such as correlation, clustering, etc., multivariate analysis can inform decision-making and contribute to the development of evidence-based policies and interventions.

Correlation refers to the degree to which two or more variables are related to each other. Correlation is useful in data analysis because it can help identify patterns and relationships between variables:

3.1 Correlation_1

```
top_crime_types = df["typeofincident"].value_counts().nlargest(5).index.tolist()
offenses_by_neighborhood = df[df["typeofincident"].isin(top_crime_types)].pivot_table(index="victimage", columns="typeofincident", va

offenses_by_neighborhood = offenses_by_neighborhood.sort_index(ascending=True)  # sort by Victim Age
sns.set(font_scale=0.8)  # adjust font size
sns.set(rc={"figure.figsize":(5,10)})  # adjust figure size
sns.heatmap(offenses_by_neighborhood, cmap="YlGnBu")
plt.title("Offenses by Age")
plt.xlabel("Offense Category")
plt.ylabel("Victim Age")
plt.xticks(rotation=45, ha='right')  # rotate labels and align right
plt.tight_layout()  # adjust spacing
plt.show()
```

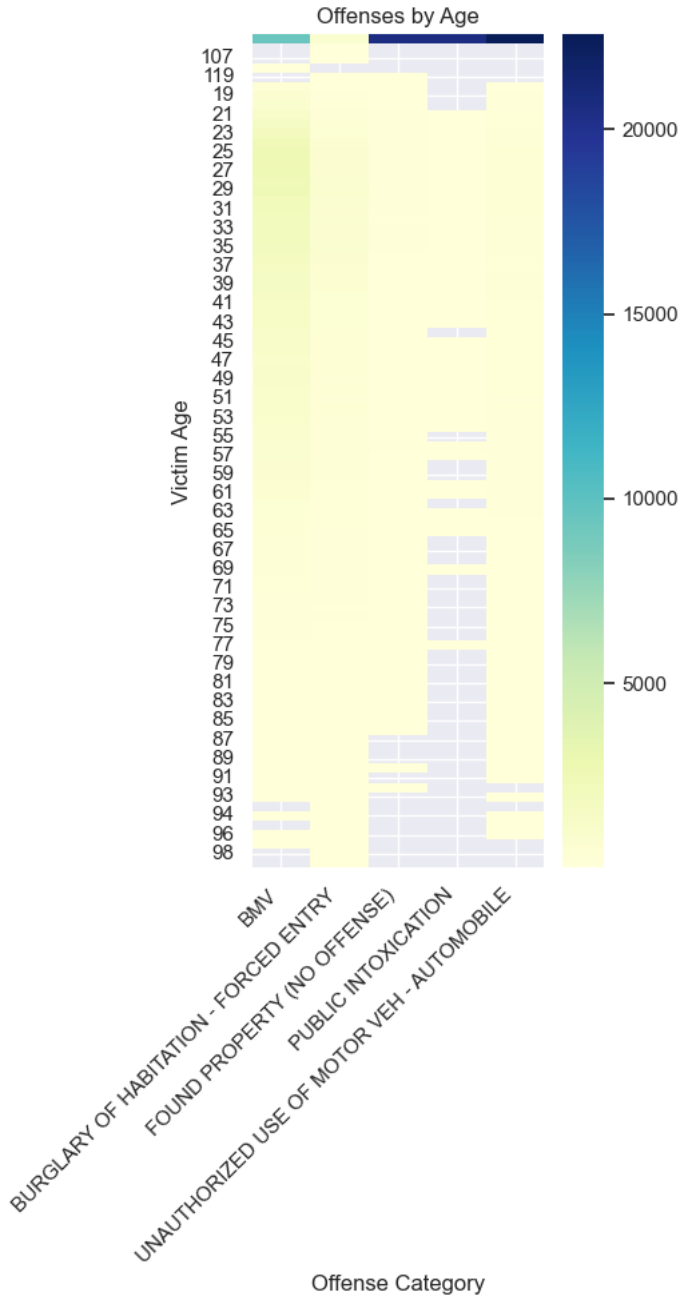

Figure 4.Relationship between Victim Age and Type of Crime

This figure shows the correlation between victim age and the type of crime. The results are straight forward that ages above 100 have high crime victims. However it is interesting to see that without above 100s, most victims in most offense categories are in between 20-40. From this relation we can provide further insight on the age group that may require more attention and resources in terms of crime prevention and victim support programs. Additionally, it may be worthwhile to investigate why there is a spike in victimization among those over 100 years old. This could potentially uncover unique challenges faced by this age group and inform the development of tailored interventions to address their needs.

3.2 Correlation_2

```
# Create a new dataframe with columns for day of week and hour of day
df_time = df[['day1oftheweek', 'time1ofoccurrence']].copy()

# Convert the time column to a datetime object
df_time['time1ofoccurrence'] = pd.to_datetime(df_time['time1ofoccurrence'], format='%H:%M')

# Set the time column as the index
df_time.set_index('time1ofoccurrence', inplace=True)

# Create a new hourly index
hourly_index = pd.date_range(start='00:00', end='23:00', freq='H')

# Pivot the data to create a 2D matrix of crime counts
df_pivot = df_time.pivot_table(index=df_time.index.time, columns='day1oftheweek', aggfunc=len)

# Reindex the pivot table to the hourly index
df_pivot = df_pivot.reindex(hourly_index.time, fill_value=0)

# Define the size of the heatmap
plt.figure(figsize=(12, 6))

# Create the heatmap
sns.heatmap(df_pivot, cmap='YlOrRd', linewidths=.5)

# Set the title and axis labels
plt.title('Crime Occurrence by Day of Week and Time of Day')
plt.xlabel('Day of Week')
plt.ylabel('Time of Day (Hourly)')

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

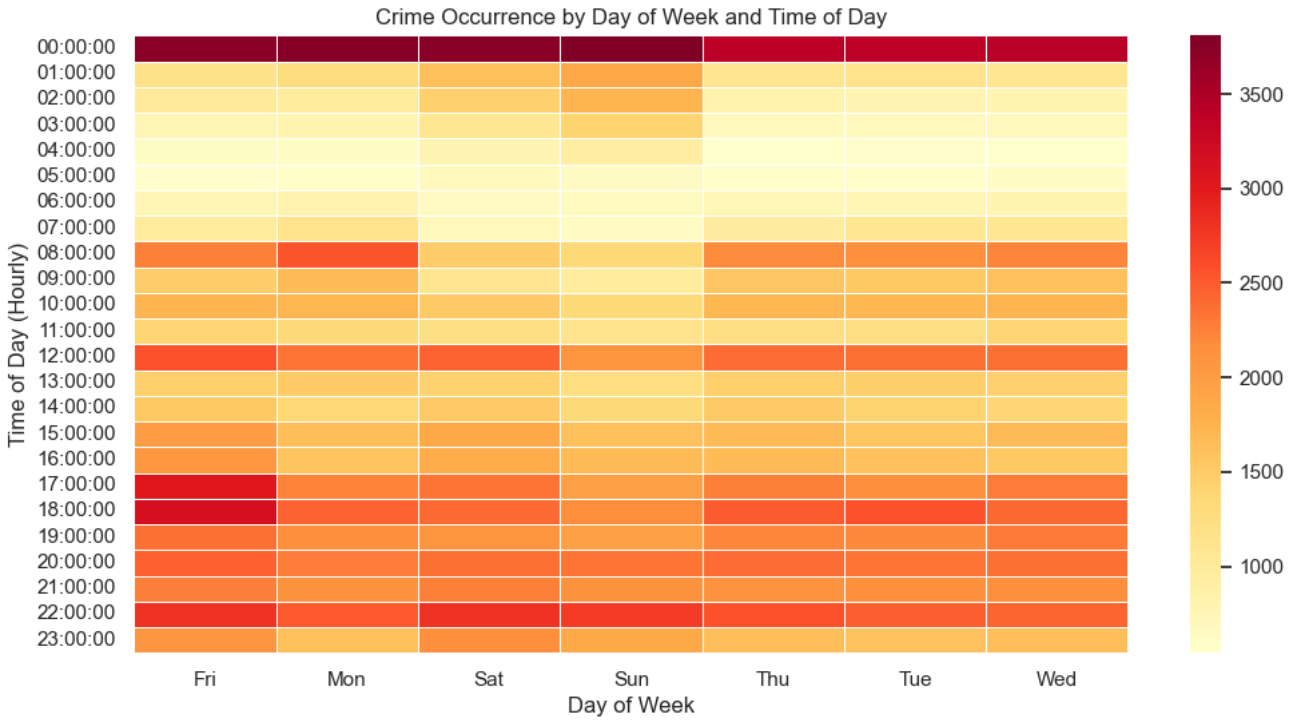

Figure 5. Relationship between Day of Week and Time of Day

Through analyzing the relationship between the day of the week and time of the day, the weakest times of the week for crime occurrence was identified. This examination of correlations between 2 variables, leads to further suggestion on inform crime prevention strategies and suggestions.

4. Suggestion

The Dallas crime data was analyzed through various stages of EDA, Univariate, and Multivariate analysis. Based on the insights obtained, a potential project idea was generated to develop a predictive model for crime occurrence in Dallas. This model would take into account various factors such as

crime type, day of the week, time of day, location, and victim information. By identifying areas with high crime rates and predicting when and where crimes are most likely to occur, law enforcement agencies can take proactive measures to prevent crime.

Further, if more intense analysis is conducted, an interactive dashboard can be developed that displays real-time crime data and allows users to filter the data by various criteria such as crime type, location, and time. This dashboard can be used by law enforcement agencies to monitor crime patterns and allocate resources accordingly, as well as by the general public to stay informed about crime in their area.

In conclusion, the project aims to provide a comprehensive understanding of crime patterns in Dallas and develop data-driven strategies to prevent and reduce crime in the city. The potential predictive model and interactive dashboard would aid in achieving this objective.