Classification Report: Predicting Crime Severity

In [413]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, r
from statsmodels.discrete.discrete_model import Logit
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
```

In [414]:

```
crime = pd.read_csv("Crime_Data_from_2020_to_Present.csv")
print(crime.shape)
```

```
(1112545, 28)
```

In [415]:

```
crime.head()
```

Out[415]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd D
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	Southwest	377	2	624	BATTEF SIMI ASSAI
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	Central	163	2	624	BATTEF SIMI ASSAI
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	Central	155	2	845	SEX OFFENI REGISTRA OUT COMPLIAN
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	N Hollywood	1543	2	745	VANDALIS MISDEAMEAN ($399 UND
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	Mission	1998	2	740	VANDALIS FELONY ($40 OVER, CHURCH V

5 rows × 28 columns

In [416]:

```
crime.columns
```

Out[416]:

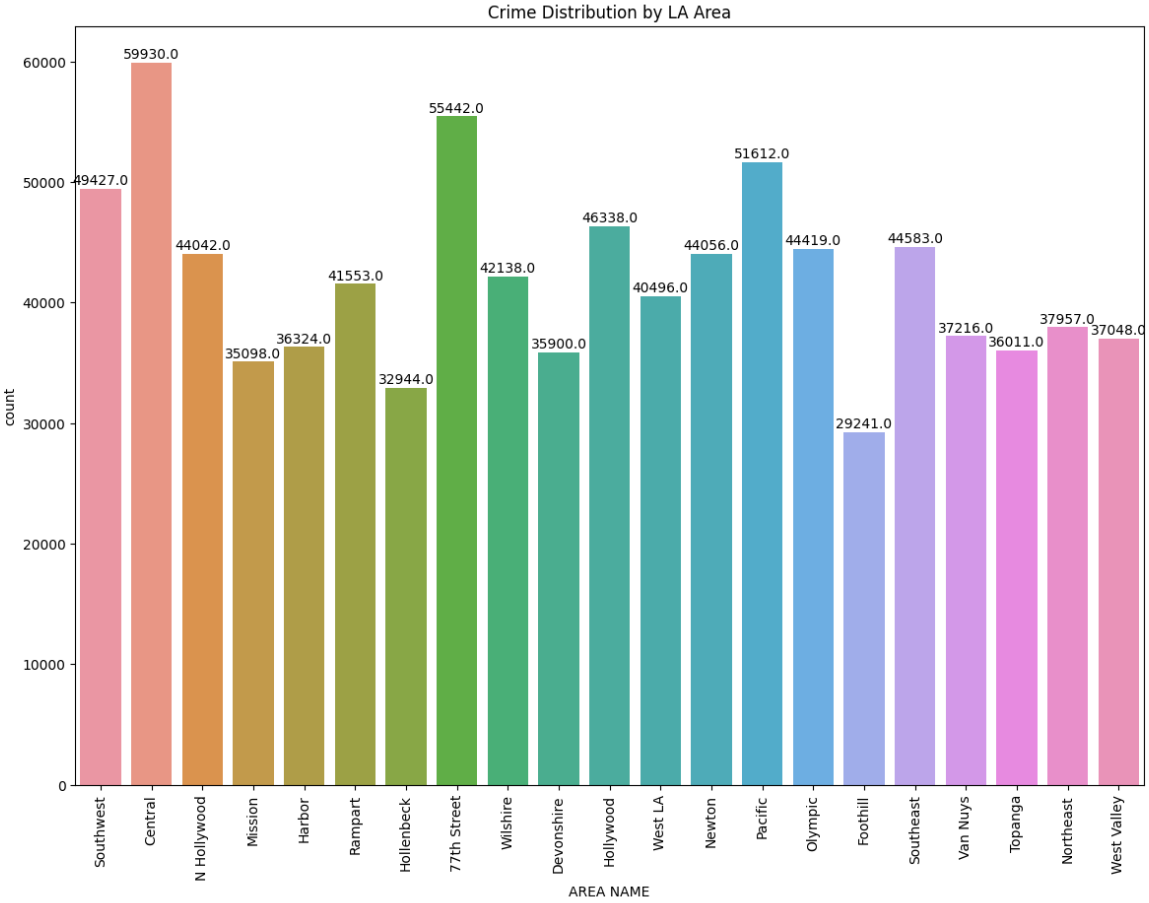
```
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA', 'AREA N
AME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocode
s',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis
Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Cr
m Cd 1',
       'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LOCATION', 'Cross Stree
t', 'LAT',
       'LON'],
      dtype='object')
```

In [417]:

```
# Data cleaning

crime = crime.drop_duplicates()
print(crime.shape)

crime['AREA'] = pd.to_numeric(crime['AREA'], errors='coerce')
print(crime['AREA'].dtype)
```

```
(881775, 28)
int64
```

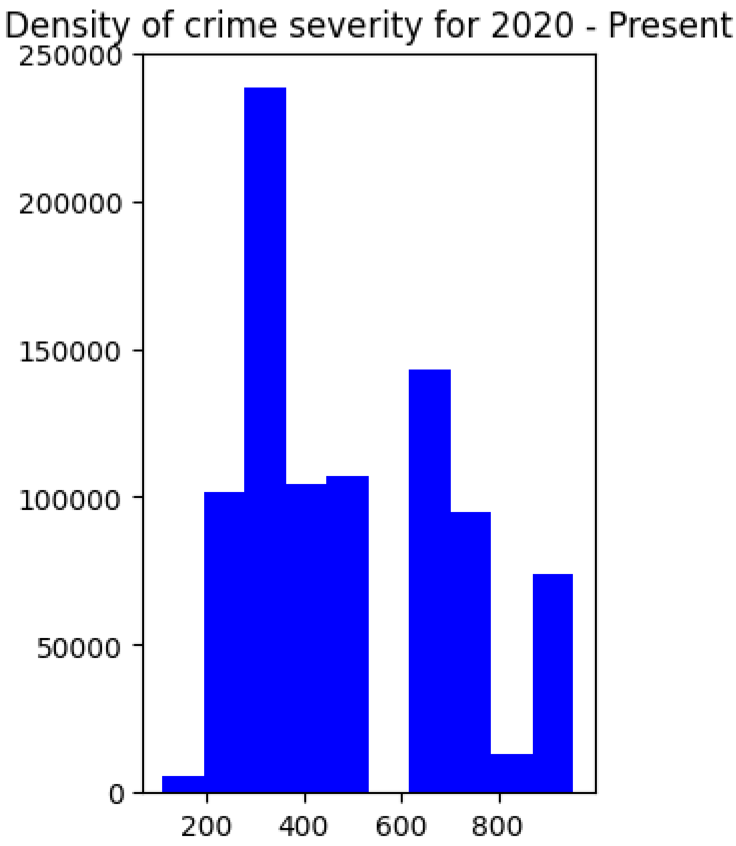
```
In [418]: # Example: Crime Distribution by LA Area
          plt.figure(figsize=(14, 10))
          ax=sns.countplot(x="AREA NAME", data=crime)
          plt.title("Crime Distribution by LA Area")
          plt.xticks(rotation=90)
          for p in ax.patches:
              ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2.,
                          ha='center', va='center', xytext=(0, 6), textcoords='c
          plt.show()
```



In [422]:

```
print(crime['Crm Cd'].describe())


plt.subplot(1, 2, 2)
plt.hist(crime['Crm Cd'], color='blue')
plt.title("Density of crime severity for 2020 - Present")
plt.show()
```

```
count    881775.000000
mean        500.990648
std         207.841176
min         110.000000
25%         331.000000
50%         442.000000
75%         626.000000
max         956.000000
Name: Crm Cd, dtype: float64
```

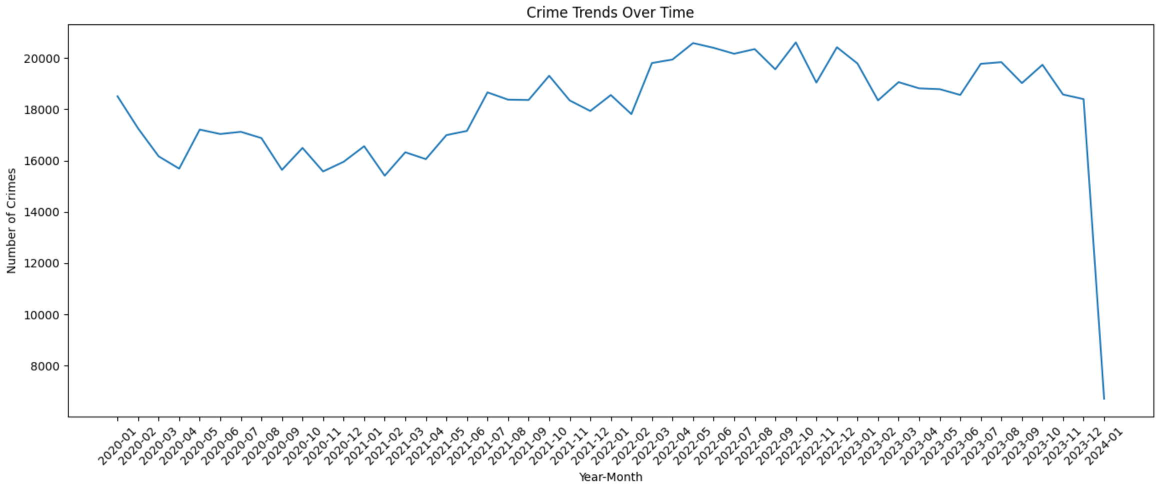


In [423]:
```
crime.columns
```

Out[423]:
```
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA', 'AREA N
AME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocode
s',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis
Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Cr
m Cd 1',
       'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LOCATION', 'Cross Stree
t', 'LAT',
       'LON'],
      dtype='object')
```

In [424]:
```
# Renaming all the columns
crime.columns = ['RecNo','ReportDate','DateOCC','TimeOCC','Area','Area
```

In [425]:
```
# Group by YearMonth and count the number of crimes
crime['DateOCC'] = pd.to_datetime(crime['DateOCC'])
crime['YearMonth'] = crime['DateOCC'].dt.to_period('M')
crime['YearMonth'] = crime['YearMonth'].astype(str)
crime
crime_trends =  crime.groupby('YearMonth').size().reset_index(name='Cr

crime_trends

# Plotting
plt.figure(figsize=(14, 6))
sns.lineplot(x='YearMonth', y='Crime Count', data=crime_trends)
plt.title('Crime Trends Over Time')
plt.xlabel('Year-Month')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [426]:

```
# Checking for null entries in our predictors
print(crime.isnull().sum())
```

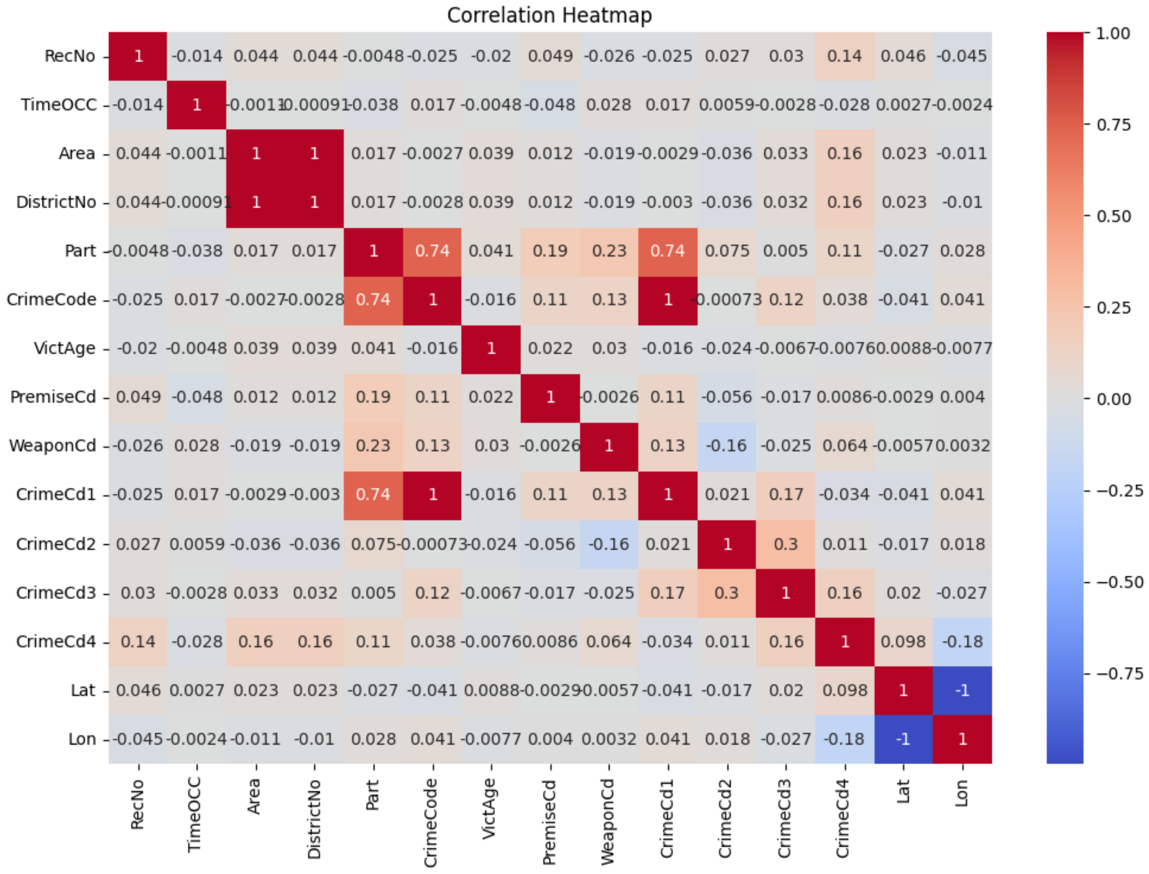
```
RecNo               0
ReportDate          0
DateOCC             0
TimeOCC             0
Area                0
AreaName            0
DistrictNo          0
Part                0
CrimeCode           0
CrmDesc             0
Mocodes        122372
VictAge             0
VictSex        116363
VictRace       116371
PremiseCd          10
PremiseDesc       537
WeaponCd       575065
WeaponDesc     575065
Status              0
StatusDesc          0
CrimeCd1           11
CrimeCd2       817200
CrimeCd3       879589
CrimeCd4       881713
Location            0
CrossStreet    742349
Lat                 0
Lon                 0
YearMonth           0
dtype: int64
```

In [427]:

```
# Removing records with null values and illogical values
crime = crime.dropna(subset=['Mocodes','VictAge','VictSex','VictRace',
crime['WeaponCd'].fillna(0, inplace=True)

print(crime.isnull().sum())
print(crime.shape)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site
-packages/pandas/core/series.py:4535: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)
  downcast=downcast,

RecNo              0
ReportDate         0
DateOCC            0
TimeOCC            0
Area               0
AreaName           0
DistrictNo         0
Part               0
CrimeCode          0
CrmDesc            0
Mocodes            0
VictAge            0
VictSex            0
VictRace           0
PremiseCd          0
PremiseDesc        0
WeaponCd           0
WeaponDesc    452773
Status             0
StatusDesc         0
CrimeCd1          10
CrimeCd2      694377
CrimeCd3      756376
CrimeCd4      758496
Location           0
CrossStreet   641453
Lat                0
Lon                0
YearMonth          0
dtype: int64
(758558, 29)
```

In [428]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(crime.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



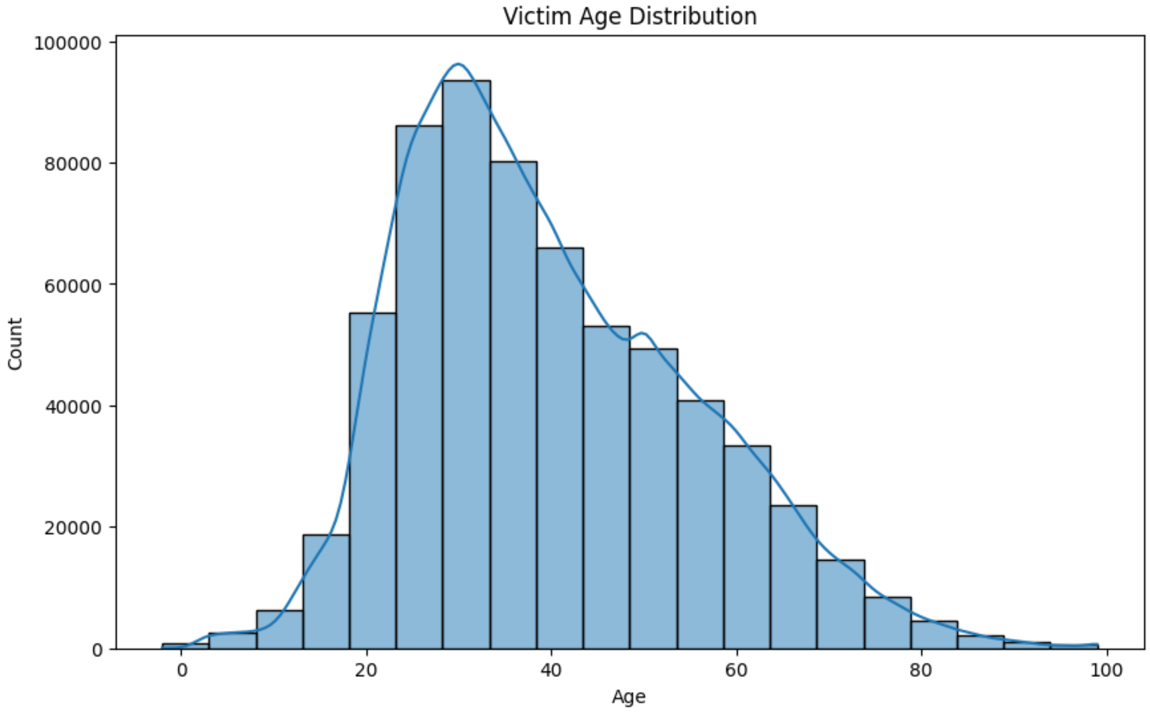
	RecNo	TimeOCC	Area	DistrictNo	Part	CrimeCode	VictAge	PremiseCd	WeaponCd	CrimeCd1	CrimeCd2	CrimeCd3	CrimeCd4	Lat	Lon
RecNo	1	-0.014	0.044	0.044	-0.0048	-0.025	-0.02	0.049	-0.026	-0.025	0.027	0.03	0.14	0.046	-0.045
TimeOCC	-0.014	1	-0.0011	-0.00091	-0.038	0.017	-0.0048	-0.048	0.028	0.017	0.0059	-0.0028	-0.028	0.0027	-0.0024
Area	0.044	-0.0011	1	1	0.017	-0.0027	0.039	0.012	-0.019	-0.0029	-0.036	0.033	0.16	0.023	-0.011
DistrictNo	0.044	-0.00091	1	1	0.017	-0.0028	0.039	0.012	-0.019	-0.003	-0.036	0.032	0.16	0.023	-0.01
Part	-0.0048	-0.038	0.017	0.017	1	0.74	0.041	0.19	0.23	0.74	0.075	0.005	0.11	-0.027	0.028
CrimeCode	-0.025	0.017	-0.0027	-0.0028	0.74	1	-0.016	0.11	0.13	1	-0.00073	0.12	0.038	-0.041	0.041
VictAge	-0.02	-0.0048	0.039	0.039	0.041	-0.016	1	0.022	0.03	-0.016	-0.024	-0.0067	-0.0076	0.0088	-0.0077
PremiseCd	0.049	-0.048	0.012	0.012	0.19	0.11	0.022	1	-0.0026	0.11	-0.056	-0.017	0.0086	-0.0029	0.004
WeaponCd	-0.026	0.028	-0.019	-0.019	0.23	0.13	0.03	-0.0026	1	0.13	-0.16	-0.025	0.064	-0.0057	0.0032
CrimeCd1	-0.025	0.017	-0.0029	-0.003	0.74	1	-0.016	0.11	0.13	1	0.021	0.17	-0.034	-0.041	0.041
CrimeCd2	0.027	0.0059	-0.036	-0.036	0.075	-0.00073	-0.024	-0.056	-0.16	0.021	1	0.3	0.011	-0.017	0.018
CrimeCd3	0.03	-0.0028	0.033	0.032	0.005	0.12	-0.0067	-0.017	-0.025	0.17	0.3	1	0.16	0.02	-0.027
CrimeCd4	0.14	-0.028	0.16	0.16	0.11	0.038	-0.0076	0.0086	0.064	-0.034	0.011	0.16	1	0.098	-0.18
Lat	0.046	0.0027	0.023	0.023	-0.027	-0.041	0.0088	-0.0029	-0.0057	-0.041	-0.017	0.02	0.098	1	-1
Lon	-0.045	-0.0024	-0.011	-0.01	0.028	0.041	-0.0077	0.004	0.0032	0.041	0.018	-0.027	-0.18	-1	1

In [429]:
```
# Removing unknown records in VictSex and VictRace, Removing 0 in Vict
crime = crime[(crime['VictSex'] != "X") & (crime['VictRace'] != "X") &
print(crime.shape)

crime = crime.drop(columns=['CrimeCd2', 'CrimeCd3', 'CrimeCd4', 'Lat',
```

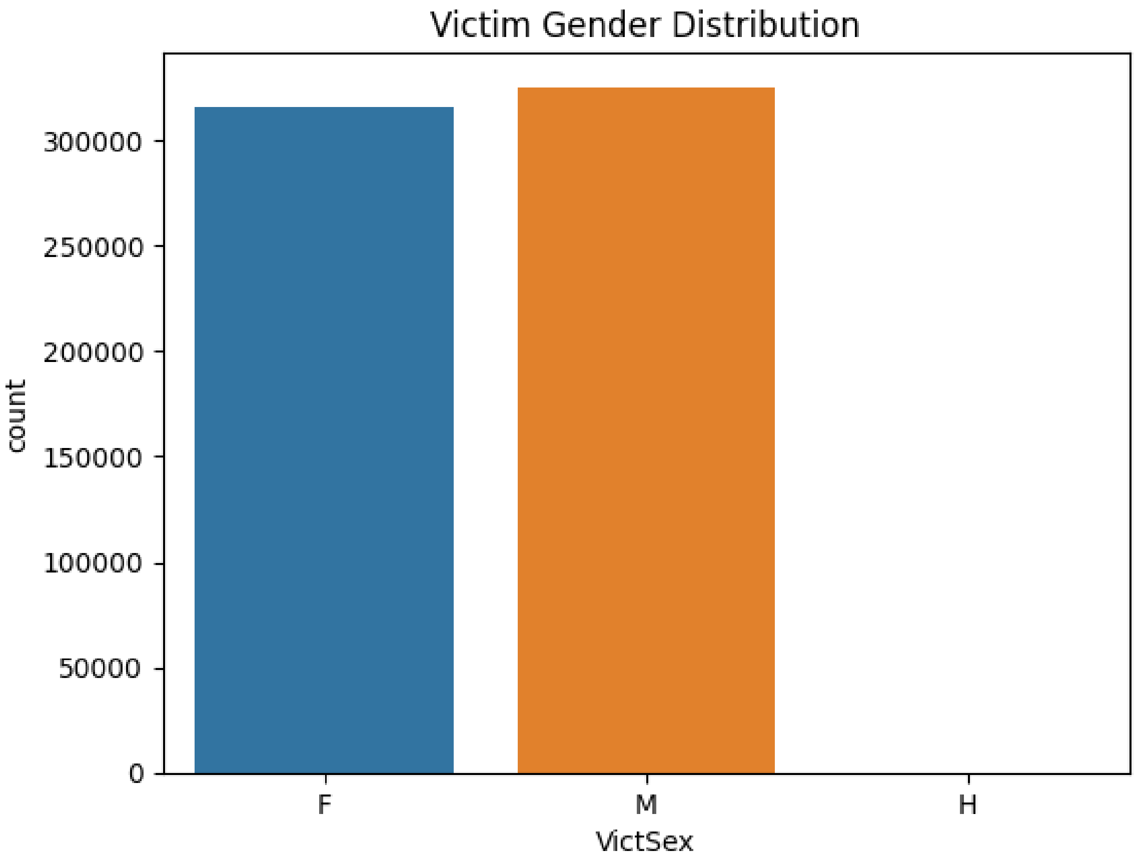
```
(641572, 29)
```

In [430]:

```
# Example: Victim Age Distribution
plt.figure(figsize=(10, 6))
sns.histplot(crime["VictAge"], bins=20, kde=True)
plt.title("Victim Age Distribution")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```



```
In [386]: sns.countplot(x='VictSex', data=crime)
          plt.title("Victim Gender Distribution")

Out[386]: Text(0.5, 1.0, 'Victim Gender Distribution')
```



```
In [431]: # Transforming columns - CrimeCode
          crime['Severity'] = np.where(crime['CrimeCode'] < 300, 'Severe', 'Non-
          crime['Severity'] = pd.Categorical(crime['Severity'])

          crime = crime.drop(columns=['CrimeCode', 'CrimeCd1'])
```

```
In [432]: # Transforming columns - VictSex and Weapon
          crime['Female'] = np.where(crime['VictSex'] == 'F', 'Yes', 'No')
          crime['Female'] = pd.Categorical(crime['Female'])

          crime['Weapon'] = np.where(crime['WeaponCd'] == 0, 'No', 'Yes')
          crime['Weapon'] = pd.Categorical(crime['Weapon'])

          crime = crime.drop(columns=['VictSex', 'WeaponCd'])
```

In [433]:
```
crime.head()
```

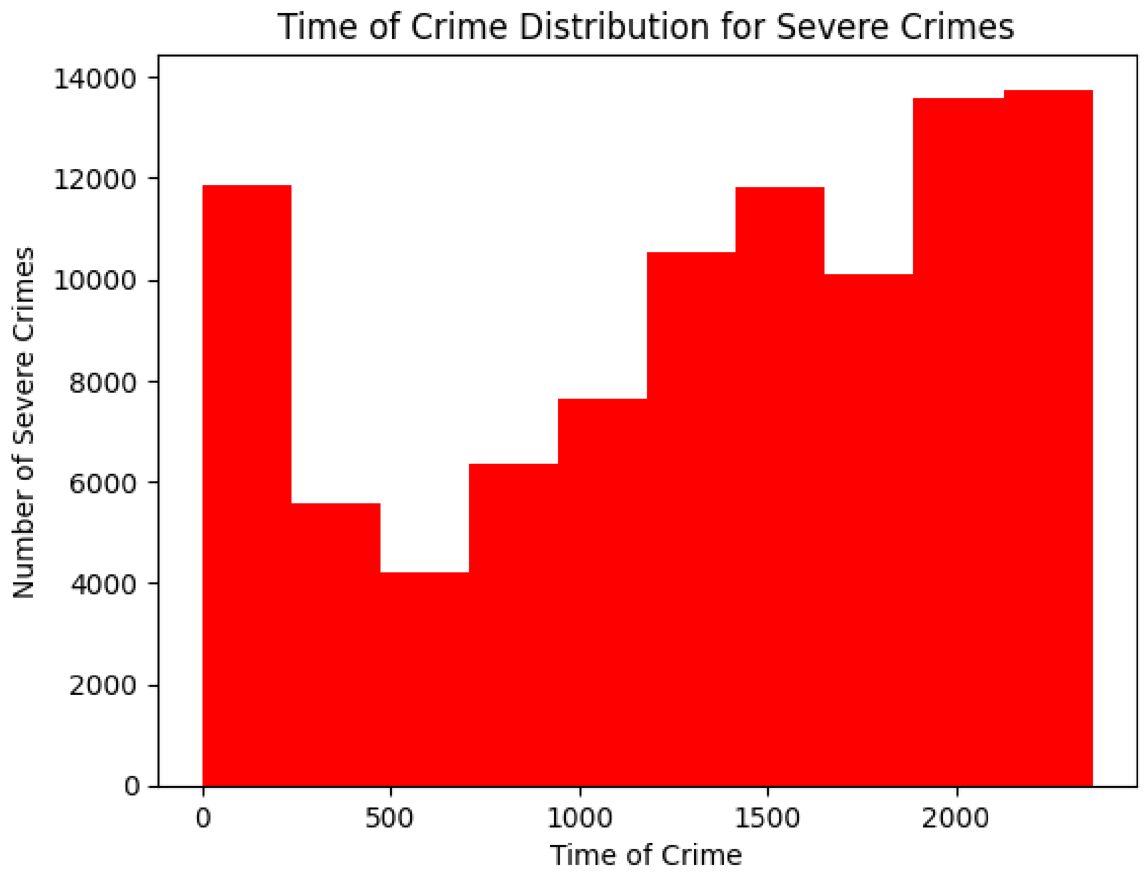
Out[433]:

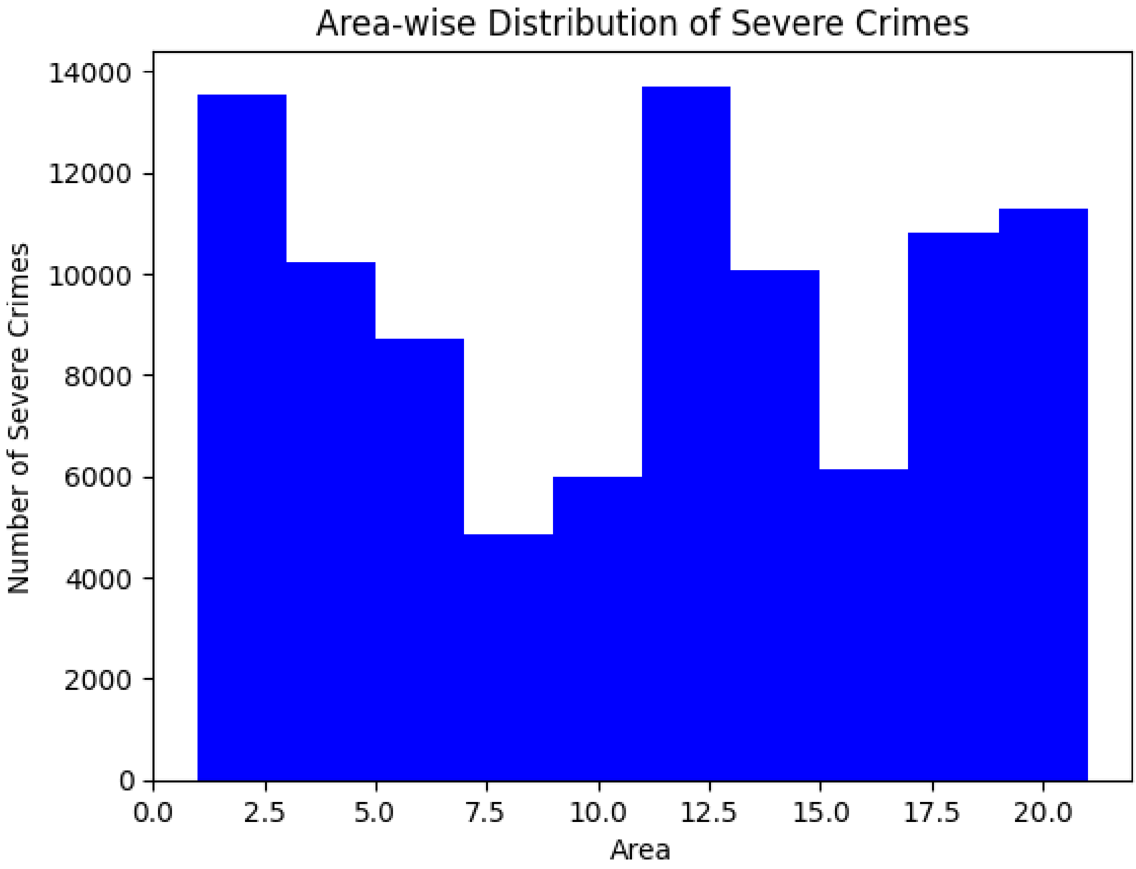
	TimeOCC	Area	VictAge	VictRace	PremiseCd	PremiseDesc	YearMonth	Severity	Female
0	2230	3	36	B	501.0	SINGLE FAMILY DWELLING	2020-01	Non-Severe	Yes
1	330	1	25	H	102.0	SIDEWALK	2020-01	Non-Severe	Nc
3	1730	15	76	W	502.0	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	2020-01	Non-Severe	Yes
5	30	1	25	H	735.0	NIGHT CLUB (OPEN EVENINGS ONLY)	2020-01	Severe	Yes
6	1315	1	23	H	404.0	DEPARTMENT STORE	2020-01	Non-Severe	Nc

In [434]:

```
# Exploratory Data Analysis
severe_exploratory = crime[crime['Severity'] == 'Severe']
severe_exploratory['TimeOCC'] = pd.to_numeric(severe_exploratory['Time

# Time
plt.hist(severe_exploratory['TimeOCC'], color='red')
plt.xlabel('Time of Crime')
plt.ylabel('Number of Severe Crimes')
plt.title('Time of Crime Distribution for Severe Crimes')
plt.show()

# Area
plt.hist(severe_exploratory['Area'], color='blue')
plt.xlabel('Area')
plt.ylabel('Number of Severe Crimes')
plt.title('Area-wise Distribution of Severe Crimes')
plt.show()
```

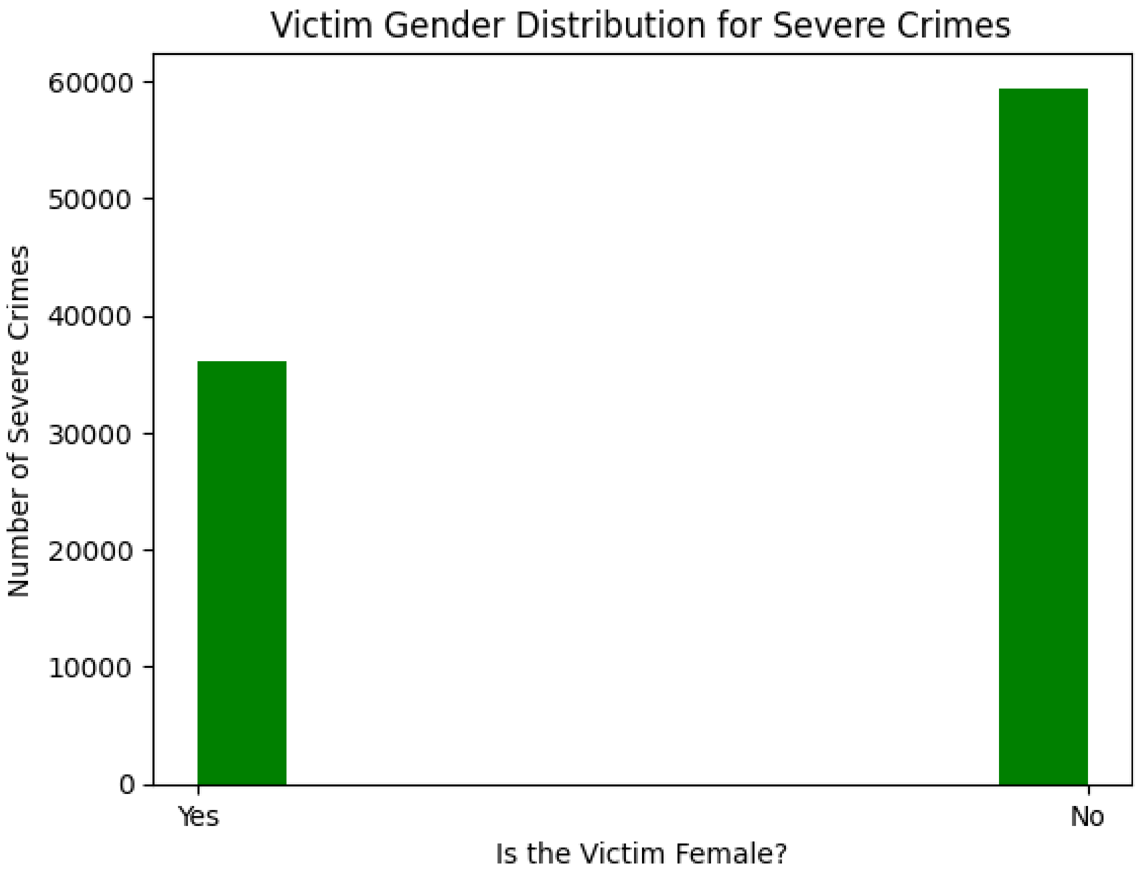
```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site
-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexin
g.html#returning-a-view-versus-a-copy)
  This is separate from the ipykernel package so we can avoid doing
imports until
```




Area-wise Distribution of Severe Crimes
Number of Severe Crimes
0
2000
4000
6000
8000
10000
12000
14000
Area
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

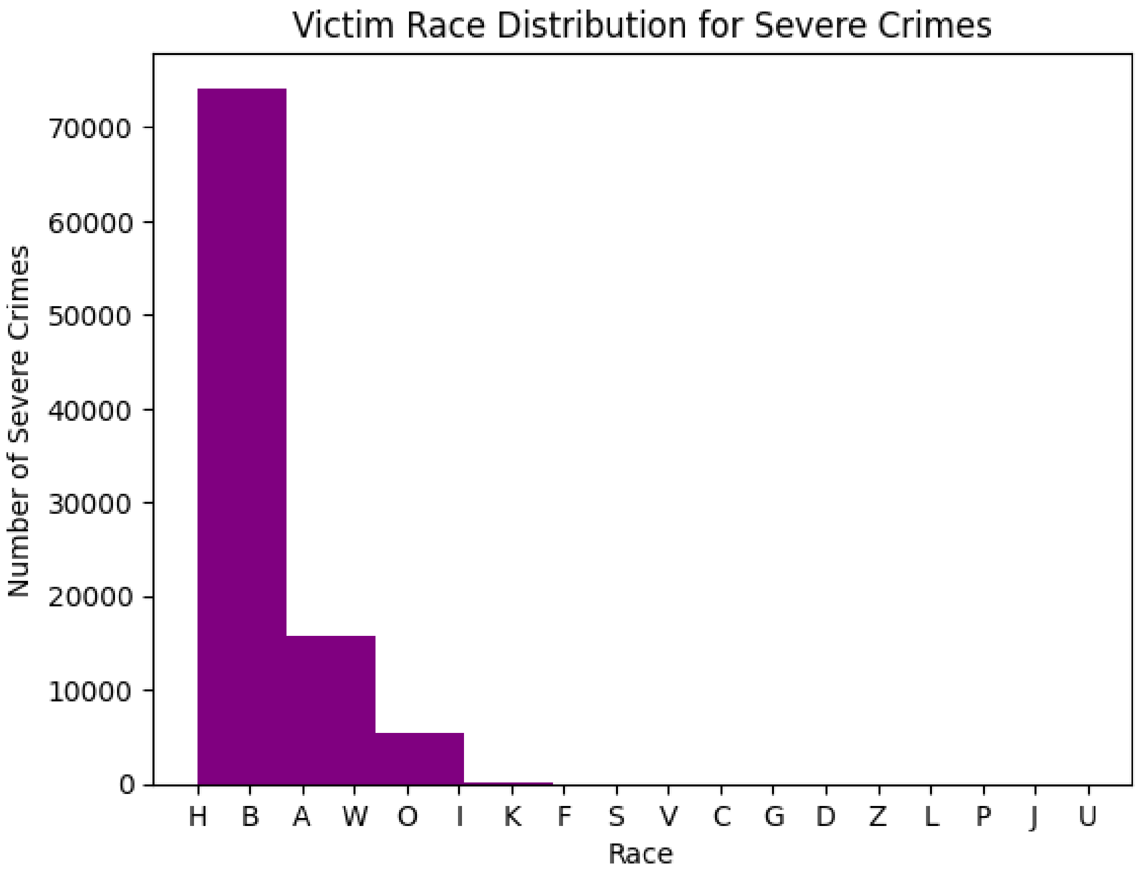
In [435]:

```
# Sex
plt.hist(severe_exploratory['Female'], color=['green'])
plt.xlabel('Is the Victim Female?')
plt.ylabel('Number of Severe Crimes')
plt.title('Victim Gender Distribution for Severe Crimes')
plt.show()
```

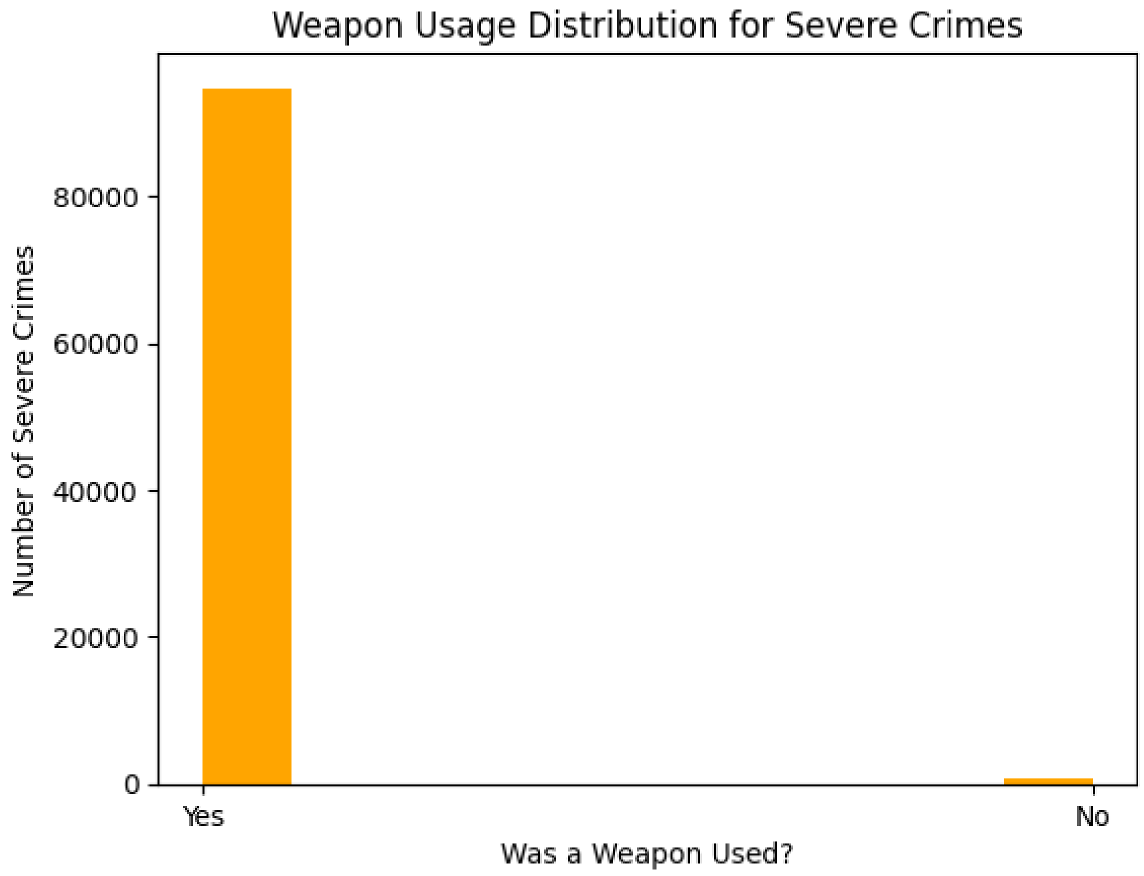


In [436]:

```
# Race
plt.hist(severe_exploratory['VictRace'], color='purple')
plt.xlabel('Race')
plt.ylabel('Number of Severe Crimes')
plt.title('Victim Race Distribution for Severe Crimes')
plt.show()
```



In [437]:
```
# Weapon
plt.hist(severe_exploratory['Weapon'], color=['orange'])
plt.xlabel('Was a Weapon Used?')
plt.ylabel('Number of Severe Crimes')
plt.title('Weapon Usage Distribution for Severe Crimes')
plt.show()
```



In [446]:
```
# Modelling
X = crime.drop(columns=['Severity', 'YearMonth'])
y = crime['Severity']

categorical_features = ['Area', 'VictRace', 'PremiseDesc', 'Female', '
numeric_features = ['TimeOCC', 'VictAge', 'PremiseCd']
```

In [447]:
```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.

# Preprocessing: One-hot encoding categorical features and scaling num
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ],
    remainder='passthrough'
)
```

In [449]:

```
X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)

# Decision Tree model
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train_preprocessed, y_train)
dt_pred = dt_model.predict(X_test_preprocessed)
```

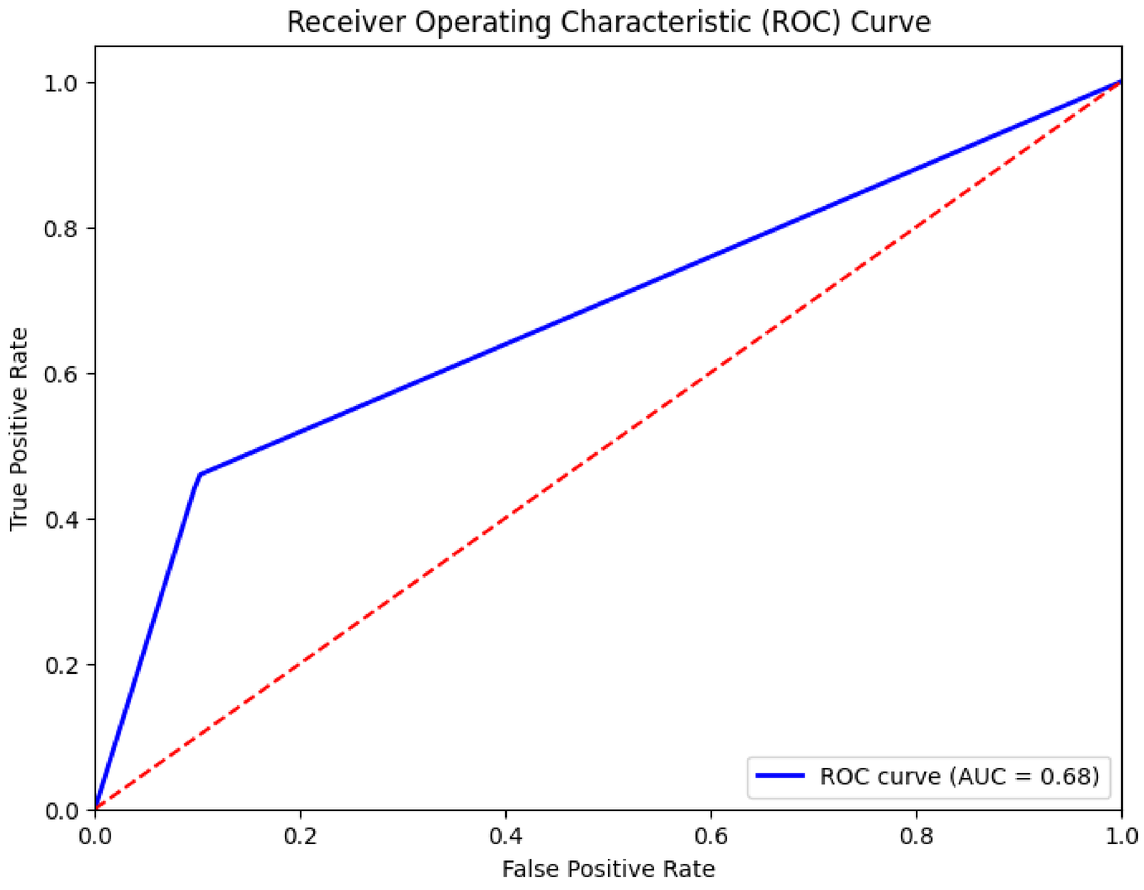
In [450]:

```
# Printing the classification report
print("Decision Tree Classifier Report:")
print(classification_report(y_test, dt_pred))
```

```
Decision Tree Classifier Report:
              precision    recall  f1-score   support

  Non-Severe       0.90      0.90      0.90    109271
      Severe       0.44      0.44      0.44     19044

    accuracy                           0.83    128315
   macro avg       0.67      0.67      0.67    128315
weighted avg       0.83      0.83      0.83    128315
```

In [451]:

```
# Predict probabilities for positive class
y_prob = dt_model.predict_proba(X_test_preprocessed)[:, 1]

# Calculate false positive rate (FPR), true positive rate (TPR), and t
fpr, tpr, thresholds = roc_curve(y_test, y_prob, pos_label='Severe')

# Calculate Area Under the Curve (AUC)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)'
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



In [452]:

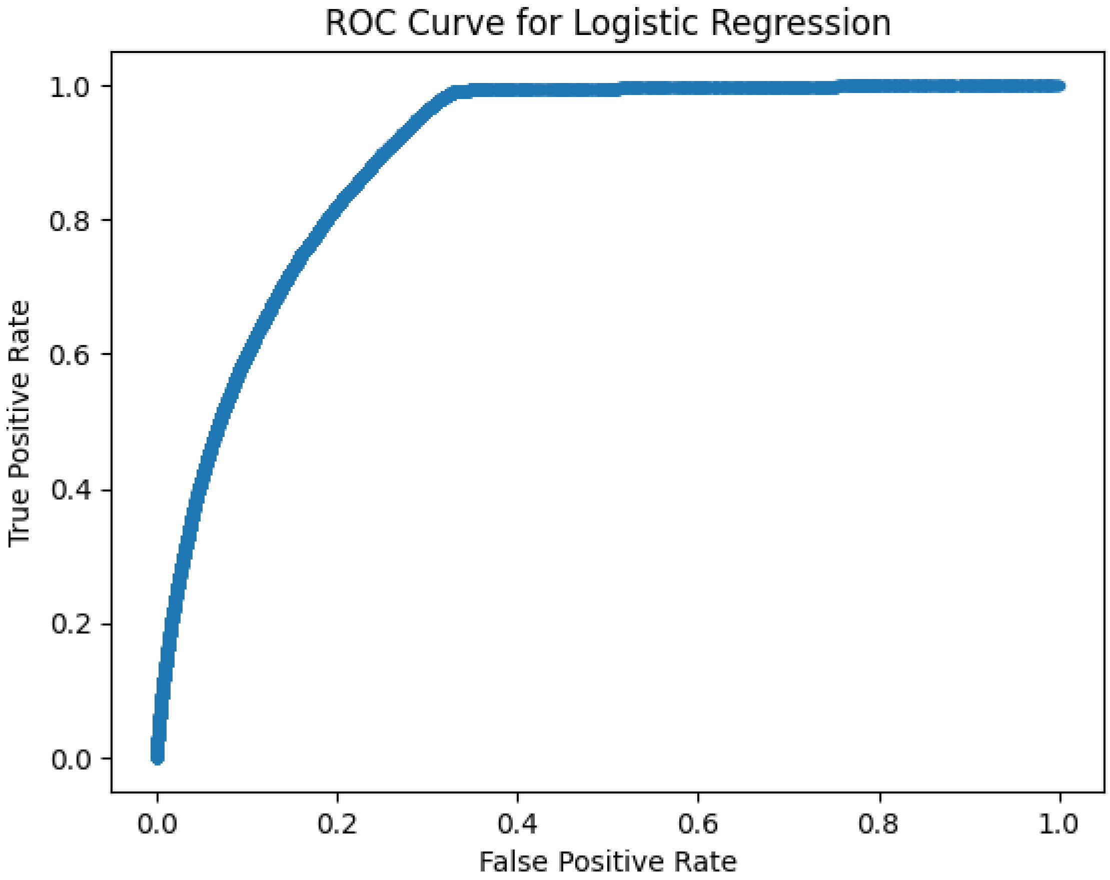
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_preprocessed, y_train)
lr_pred = lr_model.predict(X_test_preprocessed)
```

```
In [453]: print("Logistic Regression Classifier Report:")
          print(classification_report(y_test, lr_pred))
```

```
Logistic Regression Classifier Report:
              precision    recall  f1-score   support

  Non-Severe       0.90      0.96      0.93    109271
      Severe       0.61      0.38      0.46     19044

    accuracy                           0.87    128315
   macro avg       0.75      0.67      0.70    128315
weighted avg       0.86      0.87      0.86    128315
```

In [454]:

```
# Import roc_auc_score
from sklearn.metrics import roc_auc_score

# Convert categorical labels to binary labels
y_test_binary = y_test.map({'Non-Severe': 0, 'Severe': 1})

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test_binary, lr_probs)

# Calculate AUC
lr_auc = roc_auc_score(y_test_binary, lr_probs)

# Plot ROC curve
plt.plot(fpr, tpr, marker='.')
plt.title('ROC Curve for Logistic Regression')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print("AUC for Logistic Regression:", lr_auc)
```



```
AUC for Logistic Regression: 0.8967405408436027
```

In [ ]: