In [1]:
```
# imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Project Topic

The goal of this project is to predict prison sentence lengths for original sentences of primary charges in Cook County, Illinois, USA. The primary charge, according to Cook County, is the most severe charge, and cases are typically referred to by the primary charge.

The motivation behind this project is to explore features and uncover patterns related to prison sentencing lengths. This is an important task as the judicial system holds a lot of power, and a prison sentence can permanently impact many lives. Any unusual correlations or drivers of sentencing lengths can be quite impactful if identified.

Further studies can dive deeper into other sentencing types. It is important to note that biased sentencing or patterns will exist due to the nature of bias in the data collection, i.e., selective crime reporting.

The Linear Regression, Random Forest Regressor, and XGB models will be fitted to predict the sentence lengths in days. The Linear Regression model will serve as a baseline.

Due to the larger feature size (120+ after data cleaning and feature engineering), a Gradient Boost model was not used. However, due to the regularization and sampling ability, the XGB model was also selected for fitting.

While the RF model can typically be trained quicker than the XGB, due to the independent decision tree training capabilities, the XGB's sequential ensemble may have stronger predictive power on complex non-linear data.

While most of the features will be categorical and non-real, the SVM would also not be a great choice due to the size of the dataset (>10K rows). See the EDA and Machine Learning Notebook for more information.

In [2]:
```
df = pd.read_csv('Sentencing_20240531.csv', low_memory = False)
```

In [3]:
```
data = df.copy() # raw data copy for ease of iterative analysis
```

Data

Cook County State's Attorney Office. (2024). Sentencing [Data set]. SAOData. https://datacatalog.cookcountyil.gov/Courts/Sentencing/tg8v-tm6u/about_data

The data used in this project is from the Cook County State's Attorney Office (CCSA). The data was downloaded on 5/31/24, and the data set was last updated on 2/22/24. The CCSA publishes updates quarterly and provides a simple CSV export function at the website linked above.

The raw CSV file is 152.2 MB and contains 41 columns and 294,622 rows. Each row represents a separate charge.

Each column has the following data type and non-null count:

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294622 entries, 0 to 294621
Data columns (total 41 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   CASE_ID                           294622 non-null  int64
 1   CASE_PARTICIPANT_ID               294622 non-null  int64
 2   RECEIVED_DATE                     294622 non-null  object
 3   OFFENSE_CATEGORY                  294622 non-null  object
 4   PRIMARY_CHARGE_FLAG               294622 non-null  bool
 5   CHARGE_ID                         294622 non-null  int64
 6   CHARGE_VERSION_ID                 294622 non-null  int64
 7   DISPOSITION_CHARGED_OFFENSE_TITLE 294622 non-null  object
 8   CHARGE_COUNT                      294622 non-null  int64
 9   DISPOSITION_DATE                  294622 non-null  object
 10  DISPOSITION_CHARGED_CHAPTER       294622 non-null  object
 11  DISPOSITION_CHARGED_ACT           289201 non-null  object
 12  DISPOSITION_CHARGED_SECTION       289201 non-null  object
 13  DISPOSITION_CHARGED_CLASS         294603 non-null  object
 14  DISPOSITION_CHARGED_AOIC          294598 non-null  object
 15  CHARGE_DISPOSITION                294622 non-null  object
 16  CHARGE_DISPOSITION_REASON         1266 non-null    object
 17  SENTENCE_JUDGE                    293880 non-null  object
 18  SENTENCE_COURT_NAME               293228 non-null  object
 19  SENTENCE_COURT_FACILITY           292380 non-null  object
 20  SENTENCE_PHASE                    294622 non-null  object
 21  SENTENCE_DATE                     294622 non-null  object
 22  SENTENCE_TYPE                     294622 non-null  object
 23  CURRENT_SENTENCE_FLAG             294622 non-null  bool
 24  COMMITMENT_TYPE                   292819 non-null  object
 25  COMMITMENT_TERM                   292786 non-null  object
 26  COMMITMENT_UNIT                   292786 non-null  object
 27  LENGTH_OF_CASE_in_Days            275283 non-null  float64
 28  AGE_AT_INCIDENT                   290336 non-null  float64
 29  RACE                              293081 non-null  object
 30  GENDER                            293659 non-null  object
 31  INCIDENT_CITY                     273193 non-null  object
 32  INCIDENT_BEGIN_DATE               291243 non-null  object
 33  INCIDENT_END_DATE                 25268 non-null   object
 34  LAW_ENFORCEMENT_AGENCY            294295 non-null  object
 35  LAW_ENFORCEMENT_UNIT              86154 non-null   object
 36  ARREST_DATE                       288278 non-null  object
 37  FELONY_REVIEW_DATE                210345 non-null  object
 38  FELONY_REVIEW_RESULT              210345 non-null  object
 39  ARRAIGNMENT_DATE                  275283 non-null  object
 40  UPDATED_OFFENSE_CATEGORY          294622 non-null  object
dtypes: bool(2), float64(2), int64(5), object(32)
memory usage: 88.2+ MB
```

The CCSA provides the following feature descriptions, scraped from table "Columns in this Dataset" found at the link above:

In [5]:
```
pd.set_option('display.max_colwidth', None) # ease of reading the descriptio

desc = pd.read_csv('Feature Descriptions.csv')
desc
```

Out[5]:

	Column Name	Description
0	CASE_ID	Internal unique identifier for each case
1	CASE_PARTICIPANT_ID	Internal unique identifier for each defendant associated with a case
2	RECEIVED_DATE	Date when Felony Review Unit received the case
3	OFFENSE_CATEGORY	Broad offense category before specific charges are filed on a case
4	PRIMARY_CHARGE_FLAG	A binary flag indicating whether this row records the most severe charge against the accused
5	CHARGE_ID	Internal unique identifier for each charge filed
6	CHARGE_VERSION_ID	Internal unique identifier for each version of a charge associated with charges filed
7	DISPOSITION_CHARGED_OFFENSE_TITLE	Specific title of the charged offense at disposition
8	CHARGE_COUNT	Number of charges associated with one defendant in one case
9	DISPOSITION_DATE	Date charge disposed
10	DISPOSITION_CHARGED_CHAPTER	Legal Chapter for the charge at disposition
11	DISPOSITION_CHARGED_ACT	Legal Act for the charge at disposition
12	DISPOSITION_CHARGED_SECTION	Legal Section for the charge at disposition
13	DISPOSITION_CHARGED_CLASS	Legal Class for the charge at disposition
14	DISPOSITION_CHARGED_AOIC	Administrative Office of the Illinois Courts ID for law of the charge at disposition
15	CHARGE_DISPOSITION	Result of the charge
16	CHARGE_DISPOSITION_REASON	Additional information about the result of the charge
17	SENTENCE_JUDGE	Judge who oversaw the sentencing
18	SENTENCE_COURT_NAME	Circuit Court District in which the sentence was determined
19	SENTENCE_COURT_FACILITY	Courthouse in which the sentence was determined
20	SENTENCE_PHASE	When this version of the sentence was created
21	SENTENCE_DATE	Date of when the charge was sentenced
22	SENTENCE_TYPE	Broad type of sentence issued
23	CURRENT_SENTENCE_FLAG	Binary flag representing current sentence
24	COMMITMENT_TYPE	A more specific type of sentence issued
25	COMMITMENT_TERM	The number associated with the sentence
26	COMMITMENT_UNIT	Unit of sentence length
27	LENGTH_OF_CASE_in_Days	Number of days between a charge being arraigned

		and a charge being sentenced
28	AGE_AT_INCIDENT	Age of defendant at date of incident, as recorded by law enforcement or self-reported by defendant
29	RACE	Race of defendant reported by law enforcement or self-reported
30	GENDER	Gender of defendant reported by law enforcement or self-reported
31	INCIDENT_CITY	The city where the offense took place
32	INCIDENT_BEGIN_DATE	Date offense occurred/began
33	INCIDENT_END_DATE	Date offense ended
34	LAW_ENFORCEMENT_AGENCY	Law Enforcement agency associated with the arrest
35	LAW_ENFORCEMENT_UNIT	Law Enforcement Unit within Chicago Police Department associated with the arrest
36	ARREST_DATE	Date and time of arrest
37	FELONY_REVIEW_DATE	Date Felony Review result was reached
38	FELONY_REVIEW_RESULT	Result of the Felony Review process
39	ARRAIGNMENT_DATE	Date of the arraignment
40	UPDATED_OFFENSE_CATEGORY	Offense category for the case updated based upon the primary charge

Data Cleaning: Population Scope

The scope of this project was designed to study prison sentences assigned to the primary charge and for original sentences only. All other charge and cases will be dropped.

In [6]:
```
data['SENTENCE_PHASE'].value_counts()
```

Out[6]:
```
Original Sentencing                 282696
Probation Violation Sentencing        7384
Resentenced                           2121
Amended/Corrected Sentencing          2037
Remanded Sentencing                    378
Summary Charge Info                      6
Name: SENTENCE_PHASE, dtype: int64
```

In [7]:
```
data['SENTENCE_TYPE'].value_counts()
```

Out[7]:
```
Prison                                                   154023
Probation                                                112412
Jail                                                      12257
Conditional Discharge                                      4178
Supervision                                                3784
2nd Chance Probation                                       3293
Cook County Boot Camp                                      2768
Probation Terminated Unsatisfactorily                       945
Conversion                                                  277
Inpatient Mental Health Services                            275
Conditional Release                                         145
Probation Terminated Instanter                               117
Probation Terminated Satisfactorily                          77
Death                                                        70
Vocational Rehabilitation Impact Center(VRIC)                 1
Name: SENTENCE_TYPE, dtype: int64
```

In [8]:
```
data['CURRENT_SENTENCE_FLAG'].value_counts()
```

Out[8]:
```
True     284390
False     10232
Name: CURRENT_SENTENCE_FLAG, dtype: int64
```

In [9]:
```
data['PRIMARY_CHARGE_FLAG'].value_counts()
```

Out[9]:
```
True     210295
False     84327
Name: PRIMARY_CHARGE_FLAG, dtype: int64
```

In [10]:
```
data = data[(data['SENTENCE_TYPE']=='Prison')&(data['CURRENT_SENTENCE_FLAG']
            (data['PRIMARY_CHARGE_FLAG']==True)&(data['SENTENCE_PHASE']=='Or
```

Due to the nature of the data population including only prison sentences, the commitment type, according to Cook County, will most commonly be the Illinois Department of Corrections (IDC).

To simplify data modeling, the commitment type will be limited to IDC, in alignment with the Cook County glossary and guidelines.

In [11]:
```
data[['SENTENCE_TYPE','COMMITMENT_TYPE']].value_counts()
```

Out[11]:
```
SENTENCE_TYPE  COMMITMENT_TYPE
Prison         Illinois Department of Corrections          92411
               Cook County Department of Corrections         325
               Probation                                     141
               Juvenile IDOC                                  42
               710/410 Probation                              23
               Intensive Probation Services                   17
               Drug Court Probation                           12
               Cook County Impact Incarceration Program       12
               Cook County Boot Camp                          11
               Mental Health Probation                         8
               Natural Life                                    3
               Gang Probation                                  2
               Intensive Drug Probation Services               2
               Court Supervision                               1
               Drug School                                     1
               Conditional Discharge                           1
               Sex Offender Probation                          1
               Veteran's Court Probation                       1
dtype: int64
```

In [12]:
```
data = data[data['COMMITMENT_TYPE']=='Illinois Department of Corrections'].r
```

Primary Charge Flag, Sentence Type, Commitment Type, Sentence Phase, and Current Sentence Flag can be dropped as no further information can be gained.

In [13]:
```
data = data.drop(['COMMITMENT_TYPE','SENTENCE_TYPE','CURRENT_SENTENCE_FLAG',
                 'PRIMARY_CHARGE_FLAG'],
                axis = 1)
```

Data Cleaning: Data Type Munging

Looking at the .info() table above, there appears to be a few date columns that are not in datetime.

In [14]:
```
date_col = data.columns[data.columns.str.contains('_DATE', case = False, na
date_col
```

Out[14]:
```
Index(['RECEIVED_DATE', 'DISPOSITION_DATE', 'SENTENCE_DATE',
       'INCIDENT_BEGIN_DATE', 'INCIDENT_END_DATE', 'ARREST_DATE',
       'FELONY_REVIEW_DATE', 'ARRAIGNMENT_DATE'],
      dtype='object')
```

In [15]:
```
data[date_col].sample(n = 5) # random view of 5 rows
```

Out[15]:

	RECEIVED_DATE	DISPOSITION_DATE	SENTENCE_DATE	INCIDENT_BEGIN_DATE	INC
69736	07/10/2017 12:00:00 AM	11/01/2017 12:00:00 AM	11/01/2017 12:00:00 AM	07/02/2017 12:00:00 AM	
15460	10/12/2011 12:00:00 AM	02/14/2012 12:00:00 AM	02/14/2012 12:00:00 AM	10/06/2011 12:00:00 AM	
89991	03/20/2022 12:00:00 AM	06/09/2023 12:00:00 AM	06/09/2023 12:00:00 AM	01/14/2022 12:00:00 AM	
88083	06/16/2021 12:00:00 AM	06/01/2023 12:00:00 AM	06/01/2023 12:00:00 AM	06/09/2021 12:00:00 AM	
44229	06/02/2014 12:00:00 AM	07/08/2014 12:00:00 AM	07/07/2014 12:00:00 AM	06/02/2014 12:00:00 AM	

Check if the time is utilized across all 8 date fields

In [16]:
```
for col in date_col:
    print(col)
    print(data[col].str[11:].unique()) # search if there are various times f
```

```
RECEIVED_DATE
['12:00:00 AM']
DISPOSITION_DATE
['12:00:00 AM']
SENTENCE_DATE
['12:00:00 AM']
INCIDENT_BEGIN_DATE
['12:00:00 AM' nan]
INCIDENT_END_DATE
[nan '12:00:00 AM']
ARREST_DATE
['12:00:00 AM' '01:21:00 PM' '04:25:00 PM' ... '05:16:00 AM' '06:14:00 AM'
 '05:19:00 AM']
FELONY_REVIEW_DATE
[nan '12:00:00 AM']
ARRAIGNMENT_DATE
[nan '12:00:00 AM']
```

Since the time portion of the datetime does not appear to be used for all date columns except ARREST_DATE, the applicable seven date columns will be updated to a date only format.

In [17]:
```
data[date_col[date_col!='ARREST_DATE']] = data[date_col[date_col!='ARREST_DA
```

It appears that there is an erroneous DISPOSITION_DATE:

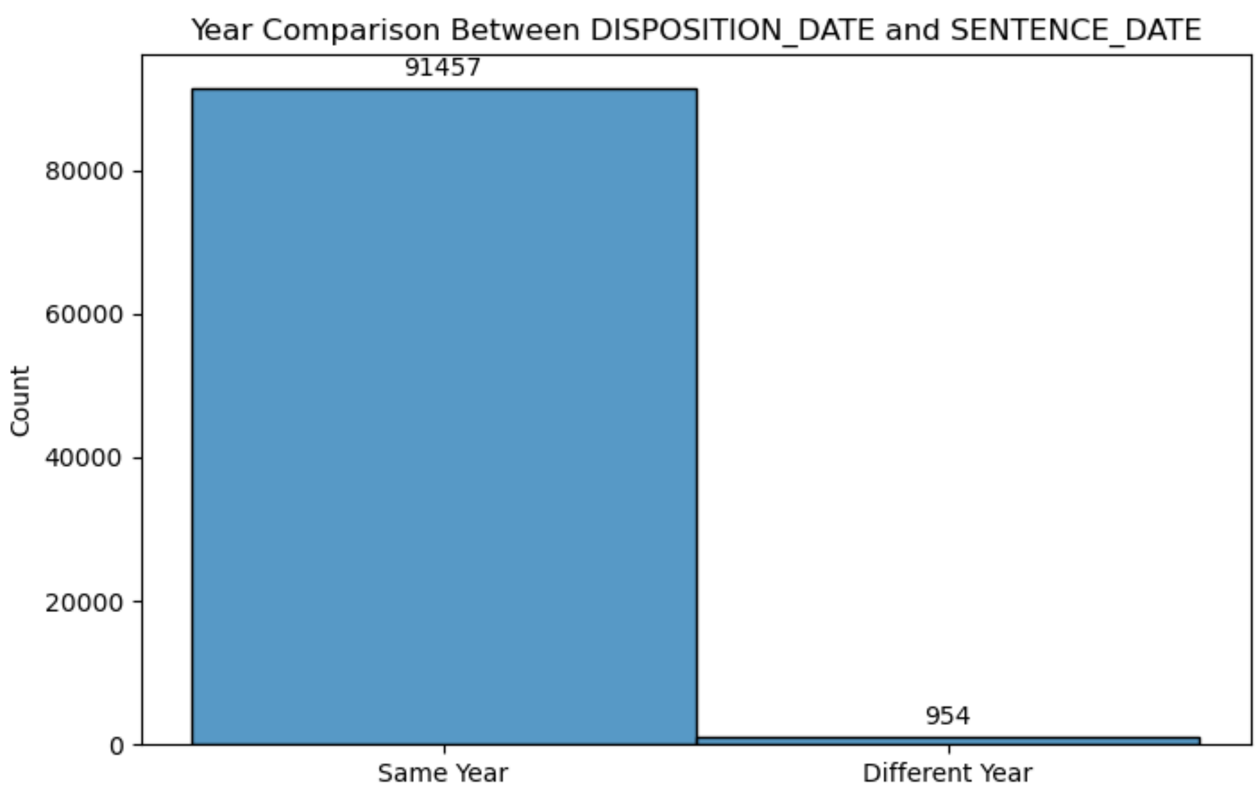
In [18]:
```
data[data['DISPOSITION_DATE'].str.contains('2912')][date_col]
```

Out[18]:

	RECEIVED_DATE	DISPOSITION_DATE	SENTENCE_DATE	INCIDENT_BEGIN_DATE	INCID
2538	01/03/2010	01/27/2912	01/27/2012	01/03/2010	

In [19]:
```
def bar_label(ph):
    for rect in ph.patches:
        height = rect.get_height()
        width = rect.get_x() + rect.get_width() / 2
        if height>=0:
            count = round(((height)),4) # Getting the label for each bar wit
        else:
            height == 0
        plt.annotate(f'{count}', xy=(width, height), xytext=(0, 3), textcoor

def year_comp_plot(date1,date2):
    plt.figure(figsize = (8,5))
    plt.title('Year Comparison Between '+str(date1)+' and '+ str(date2))
    year1 = data[date1].str[6:]
    year2 = data[date2].str[6:]
    if date1 == 'ARREST_DATE':
        year1 = data[date1].str[6:10]
    if date2 == 'ARREST_DATE':
        year2 = data[date2].str[6:10]
    plot_data = np.where(year1 == year2,'Same Year','Different Year') # plot
    plot = sns.histplot(plot_data)
    bar_label(plot)
```

In [20]:
```
year_comp_plot('DISPOSITION_DATE','SENTENCE_DATE')
```



Since almost all of the Disposition Date years are equal to the Sentence Date years, we will impute the year of the erroneous Disposition Date as 2012.

In [21]:
```
data.loc[2538,'DISPOSITION_DATE'] = '01/27/2012'
```

Upon further inspection, there appears to be more errors with the years:

- Years that are greater than 2024

In [22]:
```
future_year = {}
for col in date_col:
    if col != 'ARREST_DATE':
        print(col)
        index = data[col].dropna()[(data[col].str[6:].dropna().astype(int)>2
        print(str(len(index))+' Years > 2024') # print the number of cases w
        future_year[col] = index
    else:
        print(col)
        index = data[col].dropna()[(data[col].str[6:10].dropna().astype(int)
        print(str(len(index))+' Years > 2024')
        future_year[col] = index
```

```
RECEIVED_DATE
0 Years > 2024
DISPOSITION_DATE
5 Years > 2024
SENTENCE_DATE
21 Years > 2024
INCIDENT_BEGIN_DATE
0 Years > 2024
INCIDENT_END_DATE
0 Years > 2024
ARREST_DATE
0 Years > 2024
FELONY_REVIEW_DATE
0 Years > 2024
ARRAIGNMENT_DATE
1 Years > 2024
```

Cleaning Disposition Dates > 2024

In [23]:
```
data.loc[future_year['DISPOSITION_DATE'],date_col]
```

Out[23]:

	RECEIVED_DATE	DISPOSITION_DATE	SENTENCE_DATE	INCIDENT_BEGIN_DATE	INC
46114	08/07/2014	02/03/2045	02/03/2015	08/03/2014	
61711	05/29/2016	01/24/2047	01/24/2017	05/27/2016	
69109	06/04/2017	11/30/2107	11/30/2017	06/04/2017	
80726	06/08/2019	12/31/2028	09/03/2021	06/08/2019	
86554	12/26/2020	07/19/2921	07/19/2021	12/26/2020	

From the graph above, it appears that the disposition dates are very close, if not the same as, the sentence dates. Since it is unlikely that all sentence years for the erroneous disposition dates are incorrect (we must still check for relative outlier dates), we will correct the disposition years using the sentence year.

In [24]:
```
data.loc[future_year['DISPOSITION_DATE'],'DISPOSITION_DATE'] = data.loc[futu
```

Cleaning Sentence Dates > 2024

In [25]:
```
data.loc[future_year['SENTENCE_DATE'],date_col]
```

Out[25]:

	RECEIVED_DATE	DISPOSITION_DATE	SENTENCE_DATE	INCIDENT_BEGIN_DATE	INC
1963	08/30/2009	09/23/2011	09/23/2201	08/29/2009	
17403	12/10/2011	06/26/2012	06/26/2212	12/10/2011	
25252	08/28/2012	08/18/2014	08/07/2914	08/27/2012	
25354	08/29/2012	10/31/2014	10/31/2914	08/28/2012	
30798	02/24/2013	05/11/2015	05/11/2051	02/23/2013	
37929	10/11/2013	06/24/2016	06/24/2026	09/30/2013	
41254	02/13/2014	09/11/2014	09/11/2914	02/12/2014	
56311	09/22/2015	08/23/2016	06/08/2026	09/20/2015	
58070	12/04/2015	04/04/2016	04/04/2216	12/04/2015	
58577	11/13/2015	10/03/2016	10/03/2026	10/30/2015	

69992	07/23/2017	05/17/2018	05/16/2108	07/23/2017
70828	09/07/2017	11/26/2018	11/26/2218	09/07/2017
72847	01/15/2018	06/17/2019	06/17/2109	01/11/2018
74306	04/06/2018	09/13/2018	09/12/2108	04/05/2018
75572	06/19/2018	01/29/2019	01/29/2029	06/19/2018
78156	12/06/2018	03/13/2019	03/13/2119	11/27/2018
80904	06/20/2019	02/07/2022	12/31/2029	06/20/2019
85622	09/12/2020	01/10/2022	01/10/2027	09/11/2020
85706	08/13/2020	06/30/2022	12/31/2032	05/30/2020
86525	12/15/2020	06/14/2023	05/31/2038	12/04/2020
92101	04/30/2023	06/09/2023	06/09/2032	04/30/2023

Just like the erroneous disposition dates, the erroneous sentence dates seem to be close to, if not identical to, the disposition dates. While all disposition dates do not appear to have an erroneous year, rows 80904 and 85706 have ambiguous month/year issues similar to those encountered during the cleaning of disposition dates. Therefore, these two rows will be dropped.

In [26]:
```
data.loc[future_year['SENTENCE_DATE'],'SENTENCE_DATE'] = data.loc[future_yea
```

In [27]:
```
data = data.drop([80904,85706])
```

Cleaning Arraignment Date > 2024

In [28]:
```
data.loc[future_year['ARRAIGNMENT_DATE'],date_col]
```

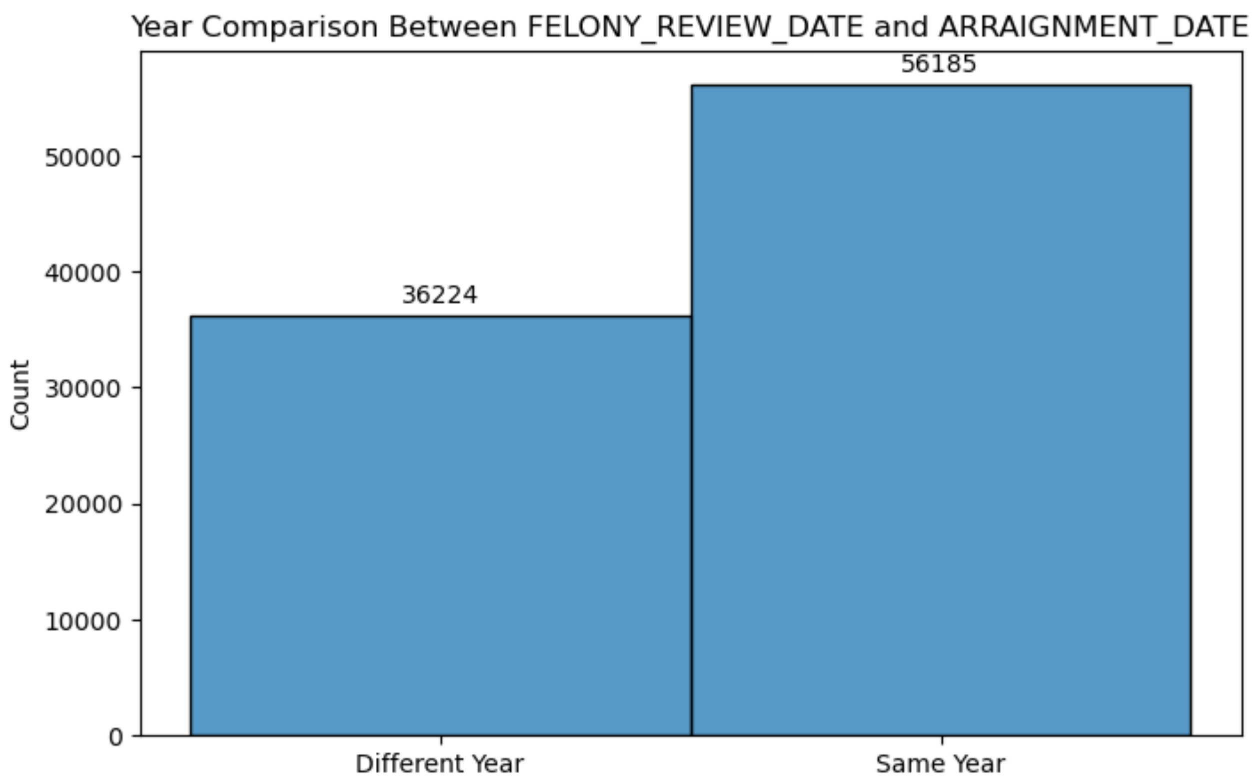
Out[28]:

	RECEIVED_DATE	DISPOSITION_DATE	SENTENCE_DATE	INCIDENT_BEGIN_DATE	INCI
67690	03/21/2017	05/18/2017	06/08/2017	03/21/2017	

The arraignment date appears to be close to the felony review date. Due to the nature of the arraignment date, there can be a few months between the felony review date and the arraignment date. This means that felony review dates closer to the end of the year will likely have an arraignment date in a different year.

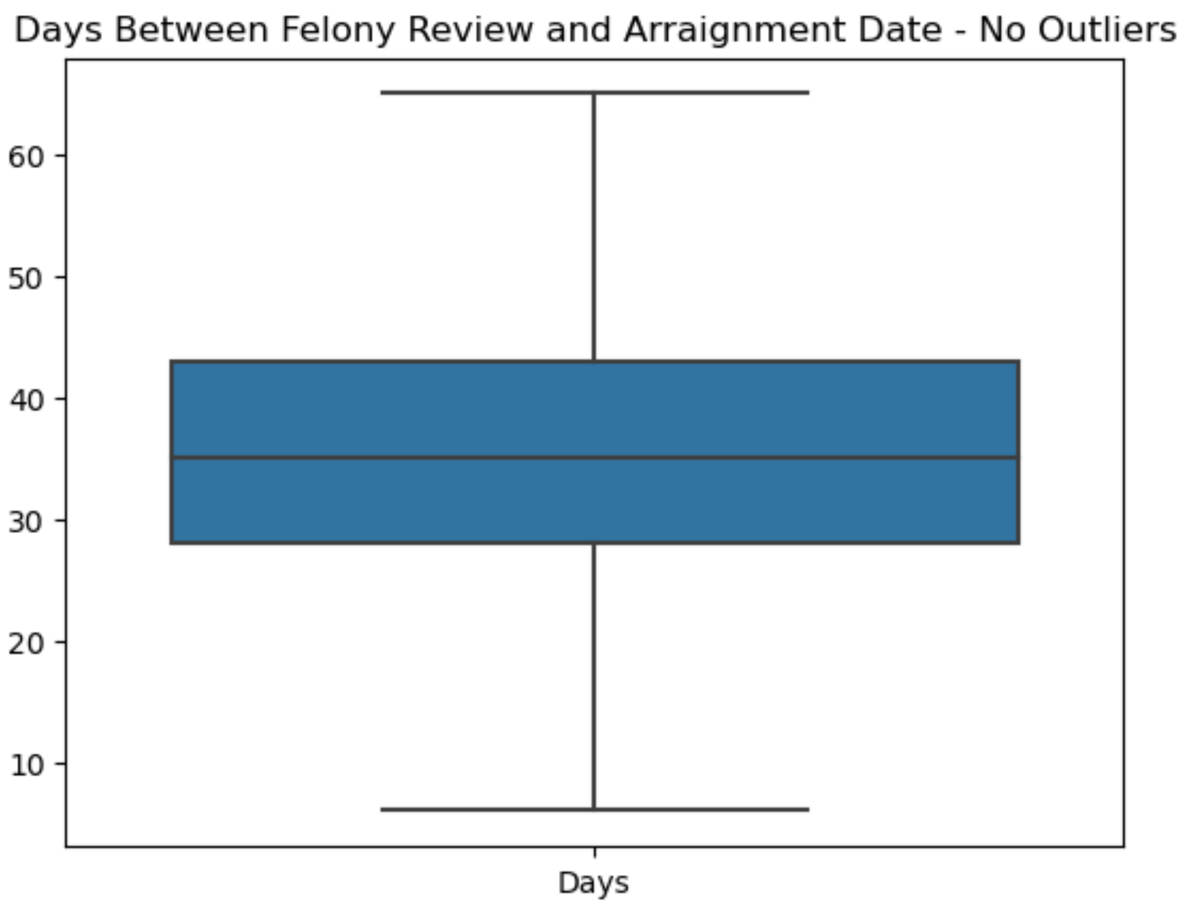
Because the felony review date is in March and the arraignment date is in April, the erroneous arraignment date will be corrected using the same felony review date year.

In [29]:
```
year_comp_plot('FELONY_REVIEW_DATE','ARRAIGNMENT_DATE')
```



In [30]:
```
plt.title('Days Between Felony Review and Arraignment Date - No Outliers')
plt_df = pd.DataFrame((pd.to_datetime(data['ARRAIGNMENT_DATE'])-pd.to_dateti
plt_df.columns = ['Days']
sns.boxplot(plt_df,showfliers = False) # hide outliers
```

Out[30]:
```
<Axes: title={'center': 'Days Between Felony Review and Arraignment Date - N
o Outliers'}>
```



In [31]:
```
data.loc[future_year['ARRAIGNMENT_DATE'],'ARRAIGNMENT_DATE'] = data.loc[futu
```

Assigning proper data types to the date columns

In [32]:
```
for col in date_col:
    data[col] = pd.to_datetime(data[col])
```

Commitment term also appears to be in an incorrect type

In [33]:
```
data['COMMITMENT_TERM'] = data['COMMITMENT_TERM'].astype(float)
```

Data Cleaning: Null Values

Columns missing a lot of values (a high percentage of null values) are difficult to impute. However, if these columns represent potentially important information, the null values must be cleaned to be used for further analysis.

Columns that are not missing many values can be imputed much more easily. Similar rows, defined by matching values in other features, can be used to assign missing values.

In [34]:
```
null_vals = pd.concat([data.isnull().sum().sort_values(ascending = False)/le
              data.isnull().sum().sort_values(ascending = False)], axis = 1)
null_vals.columns = ['Percent Null','Null Values']
null_vals = null_vals[null_vals['Percent Null']>0]
null_vals
```

Out[34]:

	Percent Null	Null Values
CHARGE_DISPOSITION_REASON	0.999448	92358
INCIDENT_END_DATE	0.918385	84867
LAW_ENFORCEMENT_UNIT	0.683267	63140
FELONY_REVIEW_RESULT	0.279367	25816
FELONY_REVIEW_DATE	0.279367	25816
INCIDENT_CITY	0.068262	6308
ARRAIGNMENT_DATE	0.054616	5047
LENGTH_OF_CASE_in_Days	0.054616	5047
ARREST_DATE	0.027465	2538
DISPOSITION_CHARGED_ACT	0.017877	1652
DISPOSITION_CHARGED_SECTION	0.017877	1652
AGE_AT_INCIDENT	0.013559	1253
INCIDENT_BEGIN_DATE	0.010497	970
SENTENCE_COURT_FACILITY	0.005984	553
SENTENCE_COURT_NAME	0.004220	390
RACE	0.002099	194
SENTENCE_JUDGE	0.001764	163
GENDER	0.001190	110
LAW_ENFORCEMENT_AGENCY	0.000725	67
DISPOSITION_CHARGED_CLASS	0.000054	5
DISPOSITION_CHARGED_AOIC	0.000043	4

The column 'CHARGE_DISPOSITION_REASON' [Additional information about the result of the charge] has over 99% null values.

The high volume of null values may not be enough to drop a column on its own. The feature description shows that this column provides additional information to 'CHARGE_DISPOSITION'.

In [35]:

```
data[data['CHARGE_DISPOSITION_REASON'].notnull()][['CHARGE_DISPOSITION',
                                                   'CHARGE_DISPOSITION_REASC
```

Out[35]:
```
CHARGE_DISPOSITION  CHARGE_DISPOSITION_REASON
Nolle Prosecution   PG to Other Count/s
19
                    Proceeding on Other Count/s
11
Plea Of Guilty      Adjudicated Minor
8
Nolle Prosecution   Proceeding on Other Case/s
3
                    Judgement & Conviction Vacated
2
Case Dismissed      Drug Court Graduate
1
Finding Guilty      Adjudicated Minor
1
Nolle Prosecution   Motion to Quash Arrest & Suppress Evidence/Sustained
1
                    Nolle - AONIC
1
                    Re-Indictment
1
                    Warrant Quashed/Recalled
1
Plea Of Guilty      PG to Other Count/s
1
Verdict Guilty      Adjudicated Minor
1
dtype: int64
```

In [36]:
```
data[data['CHARGE_DISPOSITION_REASON'].notnull()][['CHARGE_ID',
                                                   'CHARGE_DISPOSITION',
                                                   'CHARGE_DISPOSITION_REASC
                                                   'COMMITMENT_TERM','COMMIT
```

Out[36]:

	CHARGE_ID	CHARGE_DISPOSITION	CHARGE_DISPOSITION_REASON	COMMIT
37	15942849940049	Nolle Prosecution	Proceeding on Other Count/s	
98	20998074887938	Nolle Prosecution	PG to Other Count/s	
243	20022813582100	Nolle Prosecution	Judgement & Conviction Vacated	
266	20131480562697	Nolle Prosecution	PG to Other Count/s	
1763	24800790943941	Nolle Prosecution	Proceeding on Other Count/s	
3261	76127371192692	Nolle Prosecution	Motion to Quash Arrest & Suppress Evidence/Sustained	
3307	76135572905303	Nolle Prosecution	PG to Other Count/s	
3666	76163593519558	Nolle Prosecution	PG to Other Count/s	
3944	76335886599380	Case Dismissed	Drug Court Graduate	
4749	76986026534277	Nolle Prosecution	Proceeding on Other Count/s	
4792	76698292636006	Nolle Prosecution	PG to Other Count/s	
5215	76999300058029	Nolle Prosecution	PG to Other Count/s	

5989	77192017458392	Nolle Prosecution	PG to Other Count/s
6164	77185654848470	Nolle Prosecution	Proceeding on Other Count/s
7869	77794169378417	Nolle Prosecution	Warrant Quashed/Recalled
8957	77911963334468	Nolle Prosecution	Proceeding on Other Count/s
15210	79540231747718	Nolle Prosecution	PG to Other Count/s
16977	79852742128794	Nolle Prosecution	Judgement & Conviction Vacated
18048	80276672433001	Nolle Prosecution	PG to Other Count/s
18825	80489117351023	Nolle Prosecution	Proceeding on Other Count/s
22963	81368334089126	Nolle Prosecution	PG to Other Count/s
24624	81807194251804	Nolle Prosecution	PG to Other Count/s
26470	82221666313612	Plea Of Guilty	PG to Other Count/s
30766	83265922527766	Nolle Prosecution	PG to Other Count/s
31856	83540394325283	Nolle Prosecution	Proceeding on Other Count/s
33627	84008588746994	Nolle Prosecution	Proceeding on Other Count/s
33880	84067405764033	Nolle Prosecution	PG to Other Count/s
37135	84874691883318	Nolle Prosecution	PG to Other Count/s
38167	85105219407277	Nolle Prosecution	Proceeding on Other Count/s
48917	87788687266827	Nolle Prosecution	Re-Indictment
54297	89455416914564	Nolle Prosecution	PG to Other Count/s
56762	89990904217878	Nolle Prosecution	PG to Other Count/s
60562	91232835413555	Nolle Prosecution	PG to Other Count/s
61668	91586777008614	Plea Of Guilty	Adjudicated Minor
67345	93422498489731	Plea Of Guilty	Adjudicated Minor
67965	93641146095703	Plea Of Guilty	Adjudicated Minor
68397	93784162031483	Nolle Prosecution	PG to Other Count/s
70456	94589209243147	Plea Of Guilty	Adjudicated Minor
70690	94679610849837	Verdict Guilty	Adjudicated Minor
71007	94784793816024	Nolle Prosecution	PG to Other Count/s
75857	96823387742785	Plea Of Guilty	Adjudicated Minor
76112	96810765329922	Nolle Prosecution	Proceeding on Other Case/s
80846	98935180241149	Nolle Prosecution	Proceeding on Other Case/s
81731	99322534105550	Plea Of Guilty	Adjudicated Minor
84683	100601110279020	Nolle Prosecution	Proceeding on Other Count/s
86016	101493463464899	Finding Guilty	Adjudicated Minor

86776	101692771935146	Plea Of Guilty	Adjudicated Minor
87304	102343300251977	Plea Of Guilty	Adjudicated Minor
88030	102418120889306	Nolle Prosecution	Proceeding on Other Count/s
89238	103244049341434	Nolle Prosecution	Nolle - AONIC
90991	104495335972526	Nolle Prosecution	Proceeding on Other Case/s

There appears to be a lot of Nolle Prosecution entries for those with a Charge Disposition Reason. According to the Cook County Data glossary, Nolle Prosecution is used when the charge presented in the row is not pursued due to other charges.

For this project, both the charge disposition and the charge disposition reason are not necessary. This is because the project goal is to identify patterns and correlations driving sentence lengths. The charge disposition typically comes after the commitment term.

In [37]:
```
data = data.drop(['CHARGE_DISPOSITION','CHARGE_DISPOSITION_REASON'], axis =
```

Removing Incident End Date and Felony Review Date and Felony Review Result

- From our analysis above, we have other important dates to consider for further feature engineering.
- The incident end date is not a column we can impute.
- The felony review date is also often close to the case received date, and not all cases go through felony review. Both columns related to the felony review can be dropped, as the nature of the felony will be identified through other features.

In [38]:
```
data = data.drop(['INCIDENT_END_DATE','FELONY_REVIEW_DATE','FELONY_REVIEW_RE
```

Removing Law Enforcement Unit

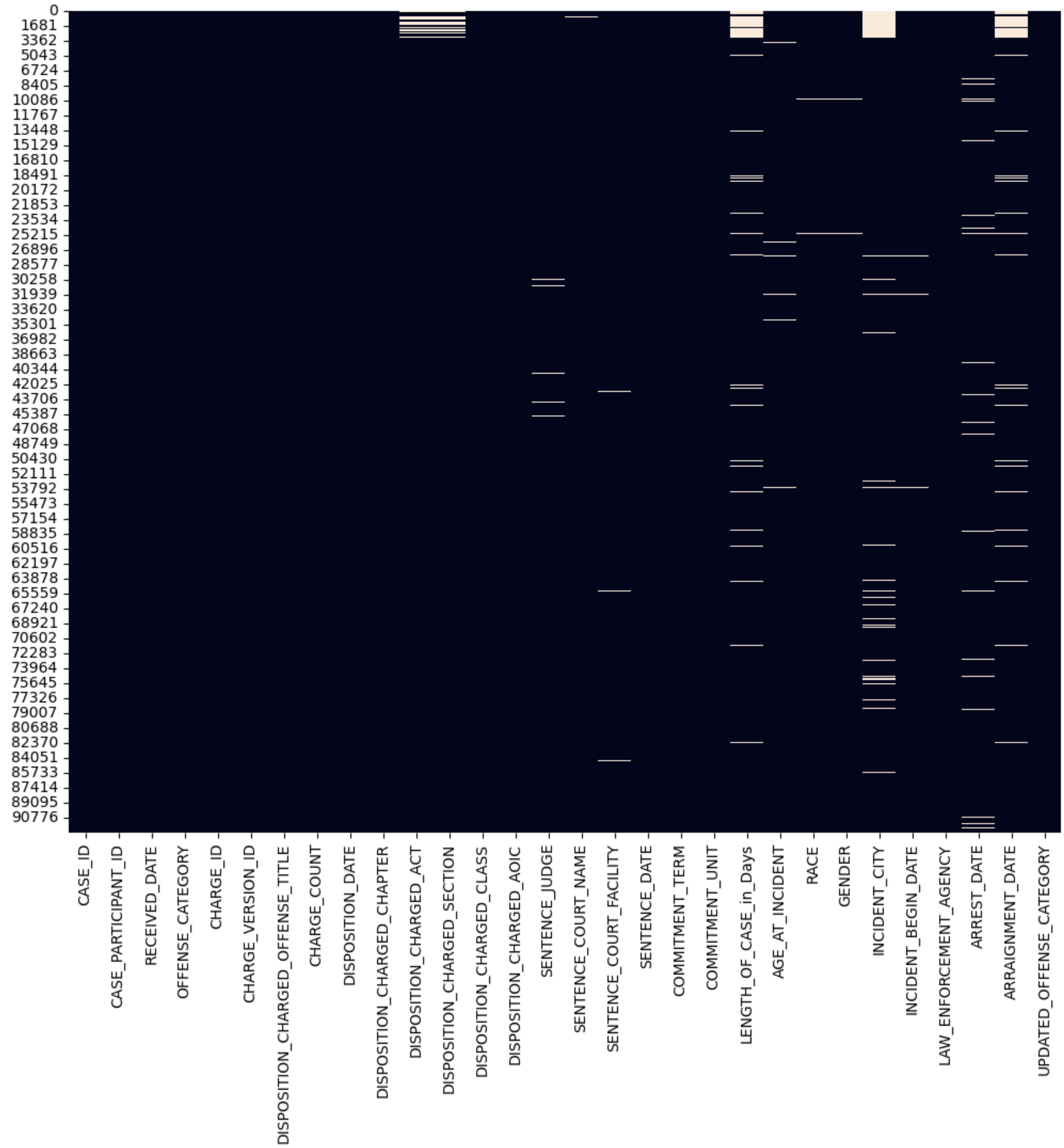
- Due to the high volume of NAs in the Law Enforcement Unit column, and most importantly, the small number of NAs in the Law Enforcement Agency column, the Unit column is dropped.

In [39]:
```
data = data.drop(['LAW_ENFORCEMENT_UNIT'], axis = 1)
```

Visualizing Remaining Null Values

In [40]:
```
plt.figure(figsize = (12,10))
sns.heatmap(data.isnull(), cbar=False, xticklabels=True) # show where in dat
```

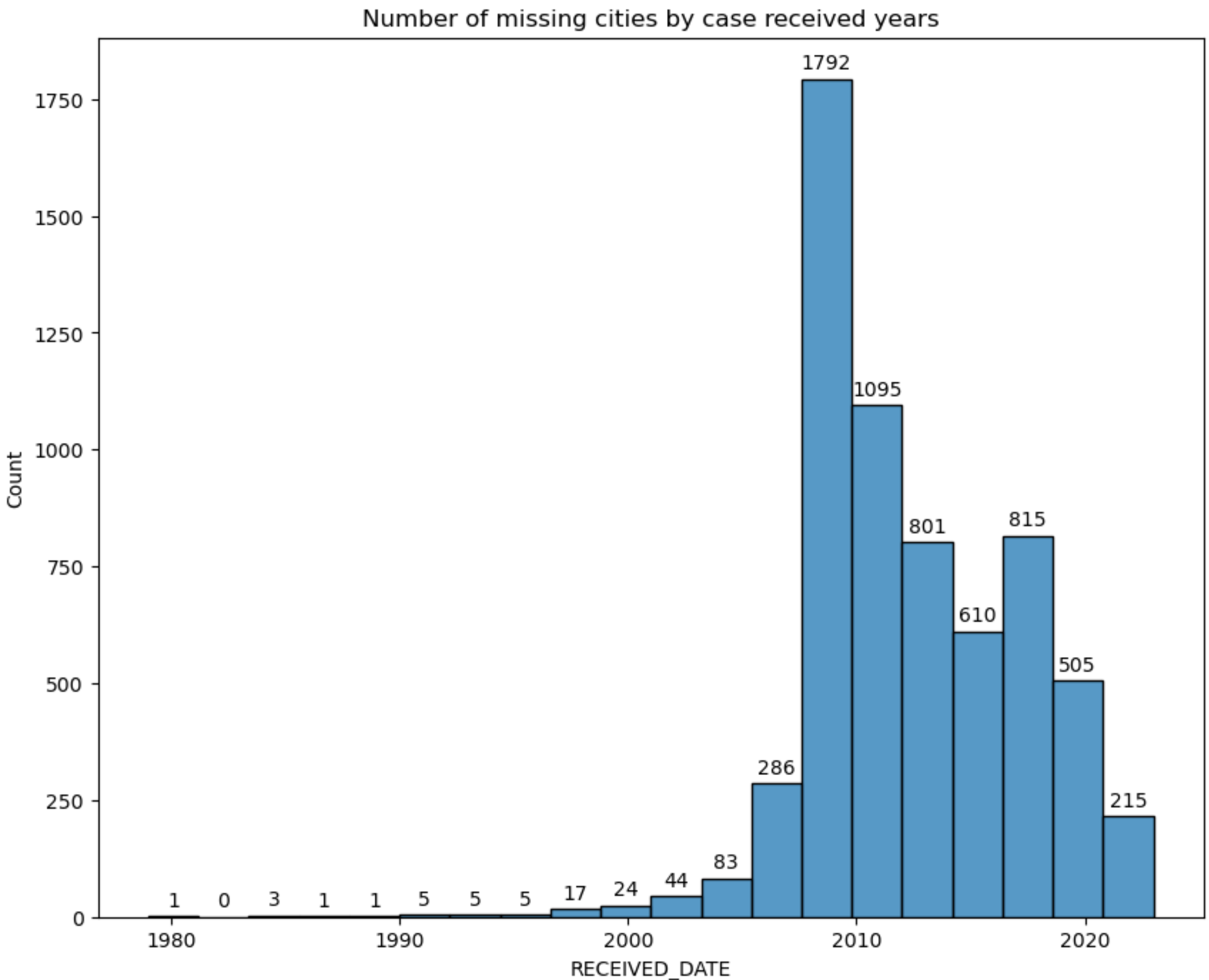
Out[40]:
```
<Axes: >
```



There appears to be a cluster of incident cities missing at the beginning of the data set. Other columns with a high density of missing values at the beginning of the data set include the disposition-related columns and the arraignment date, which in turn impacts the length of the case in days due to the interaction of these variables.

Data Cleaning: Null Values Continued

In [41]:
```
plt.figure(figsize = (10,8))
plt.title('Number of missing cities by case received years')

his_plot = sns.histplot(data[data['INCIDENT_CITY'].isnull()]['RECEIVED_DATE'
bar_label(his_plot)
```



There appears to be a few cases received in the late 1900s. These may be an outlier/mistake and could explain why the incident city was missing.

In [42]:
```
data.loc[data[data['INCIDENT_CITY'].isnull()]['RECEIVED_DATE'].idxmin(),:]
```

Out[42]:
```
CASE_ID                                                 280663370123
CASE_PARTICIPANT_ID                                     548344866796
RECEIVED_DATE                                    1979-11-20 00:00:00
OFFENSE_CATEGORY                                   Violate Bail Bond
CHARGE_ID                                             83914463242934
CHARGE_VERSION_ID                                       483380271132
DISPOSITION_CHARGED_OFFENSE_TITLE    VIO BAIL BOND/CLASS M OFFENSE
CHARGE_COUNT                                                       1
DISPOSITION_DATE                                 2016-12-16 00:00:00
DISPOSITION_CHARGED_CHAPTER                                      720
DISPOSITION_CHARGED_ACT                                            5
DISPOSITION_CHARGED_SECTION                                    32-10
DISPOSITION_CHARGED_CLASS                                          X
DISPOSITION_CHARGED_AOIC                                     1415000
SENTENCE_JUDGE                                 Lawrence Edward Flood
SENTENCE_COURT_NAME                              District 1 - Chicago
SENTENCE_COURT_FACILITY                                  26TH Street
SENTENCE_DATE                                    2016-12-16 00:00:00
COMMITMENT_TERM                                                  6.0
COMMITMENT_UNIT                                              Year(s)
LENGTH_OF_CASE_in_Days                                           NaN
AGE_AT_INCIDENT                                                 32.0
RACE                                                           Black
GENDER                                                          Male
INCIDENT_CITY                                                    NaN
INCIDENT_BEGIN_DATE                              1979-11-20 00:00:00
LAW_ENFORCEMENT_AGENCY                                    CHICAGO PD
ARREST_DATE                                                      NaT
ARRAIGNMENT_DATE                                                 NaT
UPDATED_OFFENSE_CATEGORY                           Violate Bail Bond
Name: 33238, dtype: object
```

In [43]:
```python
null_vals = pd.concat([data.isnull().sum().sort_values(ascending = False)/le
           data.isnull().sum().sort_values(ascending = False)], axis = 1)
null_vals.columns = ['Percent Null','Null Values']
null_vals = null_vals[null_vals['Percent Null']>0]
null_vals
```

Out[43]:

	Percent Null	Null Values
INCIDENT_CITY	0.068262	6308
ARRAIGNMENT_DATE	0.054616	5047
LENGTH_OF_CASE_in_Days	0.054616	5047
ARREST_DATE	0.027465	2538
DISPOSITION_CHARGED_ACT	0.017877	1652
DISPOSITION_CHARGED_SECTION	0.017877	1652
AGE_AT_INCIDENT	0.013559	1253
INCIDENT_BEGIN_DATE	0.010497	970
SENTENCE_COURT_FACILITY	0.005984	553
SENTENCE_COURT_NAME	0.004220	390
RACE	0.002099	194
SENTENCE_JUDGE	0.001764	163
GENDER	0.001190	110
LAW_ENFORCEMENT_AGENCY	0.000725	67
DISPOSITION_CHARGED_CLASS	0.000054	5
DISPOSITION_CHARGED_AOIC	0.000043	4

In [44]:

```
print('Data set without any null: '+ str(len(data.dropna())))
print('Current data set size: '+ str(len(data)))

print('\nTotal rows with at least one null: '+ str(len(data.dropna())-len(da
```

```
Data set without any null: 80581
Current data set size: 92409

Total rows with at least one null: -11828 ,-0.1279961908472118
```

Since the columns contain a small number of missing values, and/or contain values of high importance, imputing does not seem reasonable.

- Arrest date, arraignment date, age, race, sentencing judge, gender, and city are some of the columns where imputing using an estimator does not seem reasonable.
- Dropping 11,828 rows makes sense for this task.

In [45]:

```
data = data.dropna().reset_index(drop = True)
```

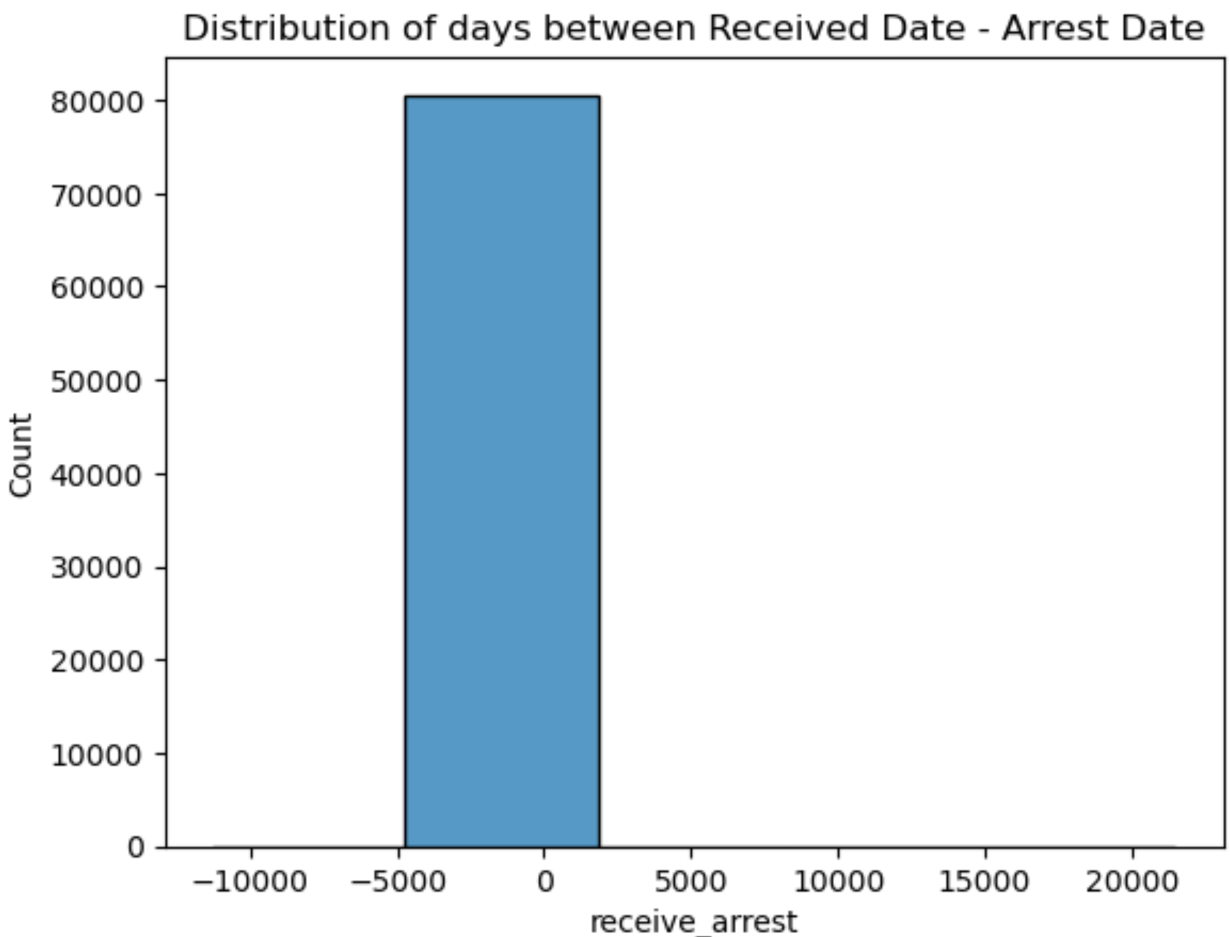
Data Cleaning: Outlier Dates

Outlier: Received - Arrest

In [46]:
```
data['receive_arrest'] = (data['RECEIVED_DATE']-data['ARREST_DATE']).dt.days
```

In [47]:
```
plt.title('Distribution of days between Received Date - Arrest Date')
sns.histplot(data, x = 'receive_arrest', bins= 5)
```

Out[47]:
```
<Axes: title={'center': 'Distribution of days between Received Date - Arrest Date'}, xlabel='receive_arrest', ylabel='Count'>
```



It appears that most of the cases have a case Received Date prior to an Arrest Date. Any cases that have an Arrest date prior to the Received date will be considered an outlier for this project and removed from the population.

In [48]:
```
data = data[data['receive_arrest']<=0]
```

Per Cook County, Arrests happen prior to Arraignments. Any cases where this is not the case will be dropped as an outlier.

In [49]:
```
print('Cases where arrests happen after the arraignment date: '+ str(sum(dat
```

```
Cases where arrests happen after the arraignment date: 85
```

In [50]:
```
data = data[data['ARRAIGNMENT_DATE']>data['ARREST_DATE']]
```

Per Cook County, Sentences happen after Arraignment. Any cases where this is not the case will be dropped as an outlier.

In [51]:
```
print('Cases where arraignments happen after the sentence date: '+ str(sum(d
```

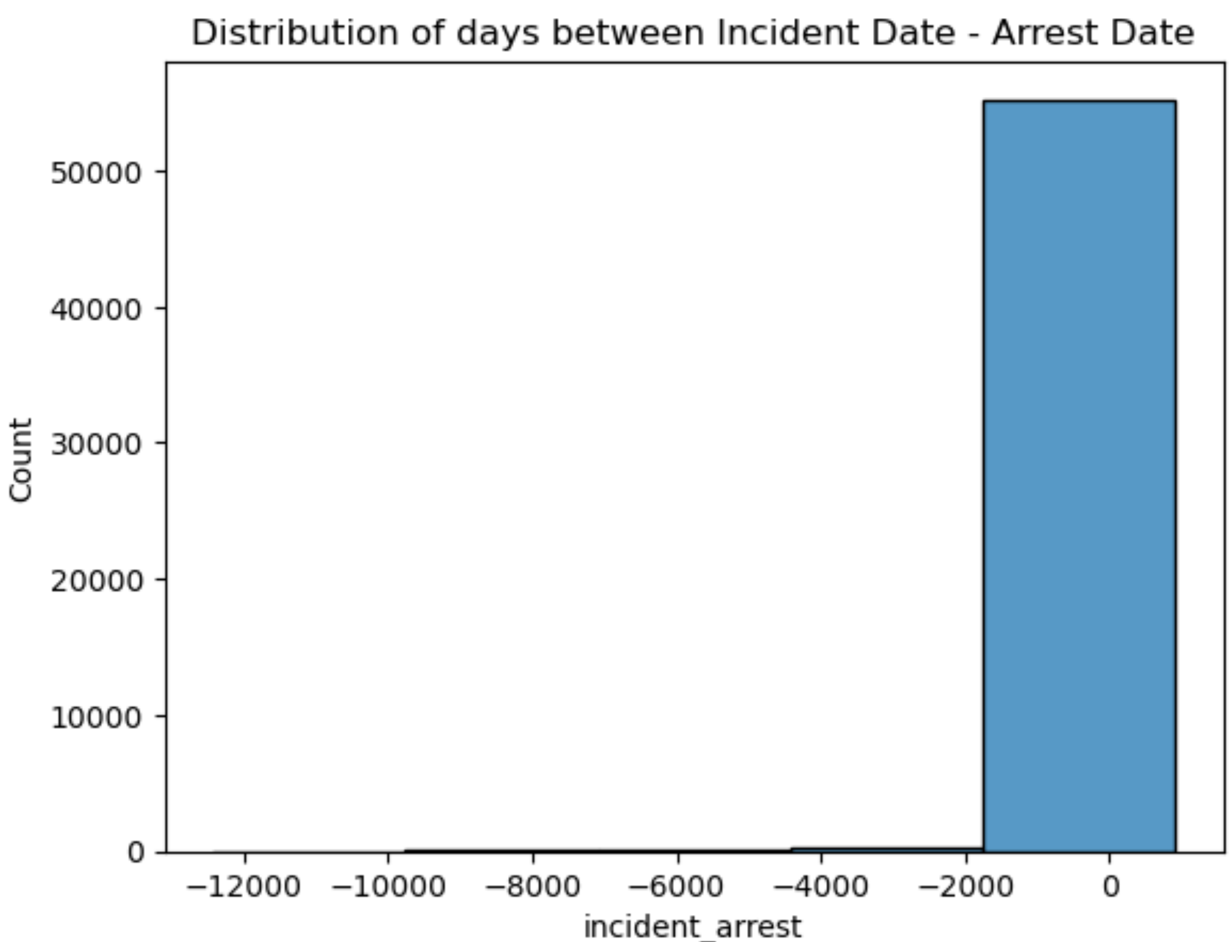
```
Cases where arraignments happen after the sentence date: 2536
```

In [52]:
```
data = data[data['SENTENCE_DATE']>data['ARRAIGNMENT_DATE']]
```

In [53]:
```
data['incident_arrest'] = (data['INCIDENT_BEGIN_DATE']-data['ARREST_DATE']).
```

In [54]:
```
plt.title('Distribution of days between Incident Date - Arrest Date')
sns.histplot(data, x = 'incident_arrest', bins= 5)
```

Out[54]:
```
<Axes: title={'center': 'Distribution of days between Incident Date - Arrest Date'}, xlabel='incident_arrest', ylabel='Count'>
```



It appears that most of the cases have a Incident Date prior to an Arrest Date. This makes sense, and any cases that have an Arrest date prior to the Incident date will be considered an outlier for this project and removed from the population.

In [55]:
```
data = data[data['incident_arrest']<=0].reset_index(drop = True)
```

In [56]:
```
data = data.drop(['receive_arrest','incident_arrest'], axis = 1)
```

Data Cleaning: Out of Scope Columns / Features

- Case Participant ID, Case ID, Charge ID, Charge Version ID:
  - These features represent the unique identifiers for the case, participant, and charge. They do not contain any further information for EDA or to be used in the machine learning model, as only the primary charge is considered.
- Length of Case in Days:
  - This feature represents the passing of time (the difference between the Arraigned Date and Sentence Date) and will be feature engineered. This pre-existing column is not needed.
- Updated Offense Category:
  - This feature represents information that evolves as the case progresses. While important, it is unclear if this "update" occurs before or after the original sentence guidelines. Therefore, for this project, it is out of scope. The Offense Category column will be kept.
- Received Date:
  - This column represents the date the State's Attorney's office first touched the case. For the purposes of this project, other more pertinent date columns will be enriched through feature engineering.
- Disposition Charged Chapter, Act, and Section:
  - According to Cook County, these three columns together represent the Illinois criminal statute of the charges brought against the participant.
- Disposition Charged Offense Title:
  - As there are over 100 unique values, the combination of Offense Category and Disposition Charged Class will be kept to provide enough detail regarding crime type and severity.

In [57]:
```
data = data.drop(['CASE_PARTICIPANT_ID','CASE_ID','CHARGE_ID','CHARGE_VERSIC
           'UPDATED_OFFENSE_CATEGORY','RECEIVED_DATE',
           'DISPOSITION_CHARGED_CHAPTER', 'DISPOSITION_CHARGED_ACT','DISPOSIT
                   'DISPOSITION_CHARGED_OFFENSE_TITLE'], axis = 1)
```

Number of unique values:

- This is used to identify good candidates for encoding categorical variables into dummy variables.
- Columns with too many unique values may not be great candidates to turn into dummy variables without further enrichment.

In [58]:
```
data.nunique().sort_values()
```

Out[58]:
```
GENDER                             5
SENTENCE_COURT_NAME                6
COMMITMENT_UNIT                    7
RACE                               9
DISPOSITION_CHARGED_CLASS         12
SENTENCE_COURT_FACILITY           12
CHARGE_COUNT                      27
AGE_AT_INCIDENT                   66
OFFENSE_CATEGORY                  84
COMMITMENT_TERM                  143
INCIDENT_CITY                    156
LAW_ENFORCEMENT_AGENCY           213
SENTENCE_JUDGE                   236
DISPOSITION_CHARGED_AOIC        1018
SENTENCE_DATE                   3253
DISPOSITION_DATE                3277
ARRAIGNMENT_DATE                3291
INCIDENT_BEGIN_DATE             5266
ARREST_DATE                    50678
dtype: int64
```

Continue dropping of out of scope columns:

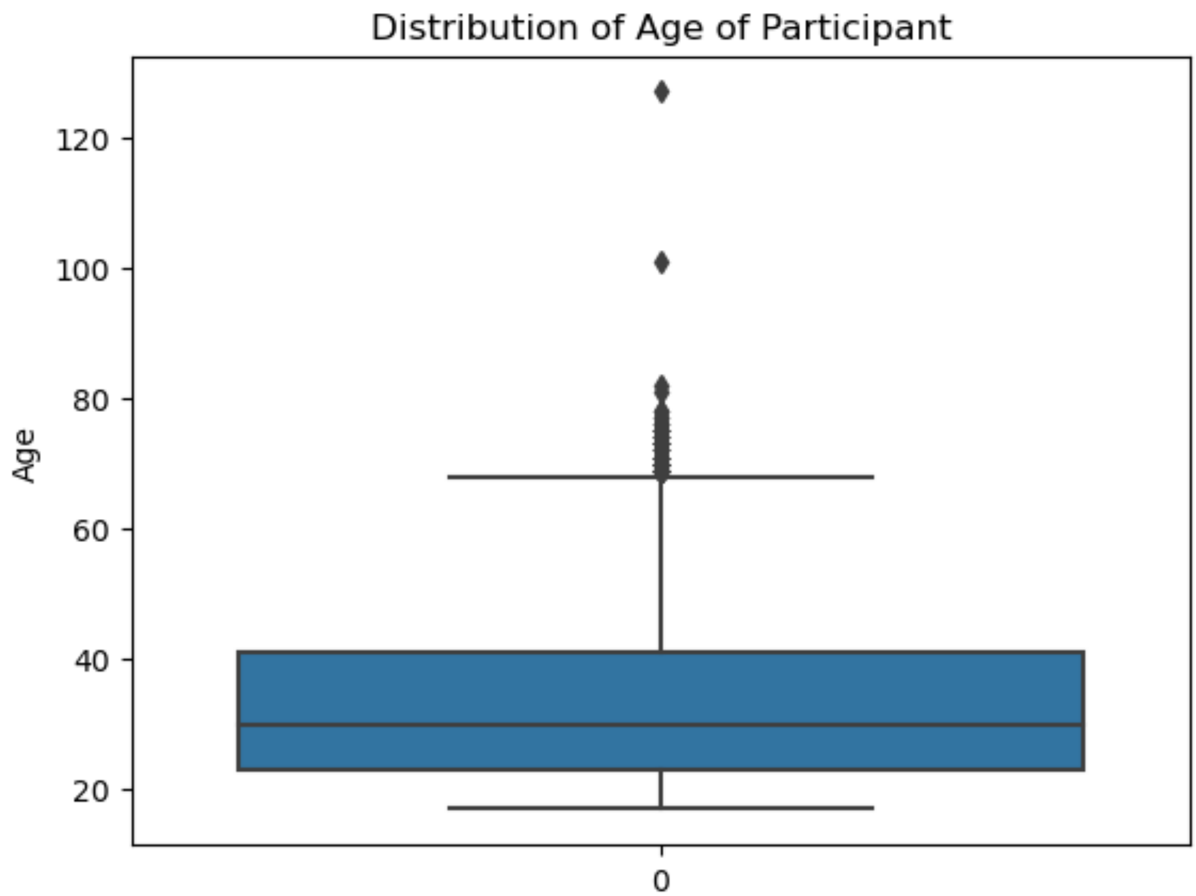
- Disposition AOIC
  - This feature represents the administrative office where the disposition was held. In this project, it will be considered out of scope due to the large number of unique offices (>1000).

In [59]:
```
data = data.drop(['DISPOSITION_CHARGED_AOIC'], axis = 1)
```

Data Cleaning: Outlier Age

In [60]:
```
plt.title('Distribution of Age of Participant')
sns.boxplot(data['AGE_AT_INCIDENT'])
plt.ylabel('Age')
```

Out[60]:
```
Text(0, 0.5, 'Age')
```



It appears highly unlikley for the two outlier participants to be over 100 years old. The oldest verified human was 122 years of age.

In [61]:
```
data[data['AGE_AT_INCIDENT']>100]
```

Out[61]:

	OFFENSE_CATEGORY	CHARGE_COUNT	DISPOSITION_DATE	DISPOSITION_CHARGED_
39562	Driving With Suspended Or Revoked License	1	2018-09-05	
52332	Aggravated Battery Police Officer	1	2022-09-21	

As age is an important feature, the two outliers are dropped due to high likelihood of error.

In [62]:
```
data = data[data['AGE_AT_INCIDENT']<100].reset_index(drop = True)
```

Feature Engineering: Time

- Time Delta Related Features:
  - arrest_incident: Arrest Date - Incident Date. This is used to identify the duration of time it took before the participant was arrested after the crime. A potential hypothesis could be that a longer duration would indicate a participant who has been "on the run", suggesting a potentially higher severity.
  - sentence_arrest: Sentence Date - Arrest Date. This is used to identify the duration of time it took before the participant was sentenced after the arrest date. A potential hypothesis could be that a longer gap would indicate a more serious case, suggesting a potentially higher severity.

In [63]:
```
date_col = data.columns[data.columns.str.contains('_DATE', case = False, na
```

In [64]:
```
data['arrest_incident'] = (data['ARREST_DATE']-data['INCIDENT_BEGIN_DATE']).
data['sentence_arrest'] = (data['SENTENCE_DATE']-data['ARREST_DATE']).dt.day
```

- Circular Encoding Months:
  - Arrest Month, Disposition Month
  - Sine and cosine circular encoding will be created to assign the correct temporal relationship between month variables. For example, month 12 is closer to month 1 than it is to month 10, but without circular encoding, the temporal relationship is not known.
  - A potential hypothesis would explore a seasonality pattern in the severity of sentences throughout the year. For instance, dispositions or arrests around holiday seasons could result in more lenient sentences.

In [65]:
```
def circular_encoding(col):
    sin = np.sin(2 * np.pi * data[col].dt.month/12)
    cos = np.cos(2 * np.pi * data[col].dt.month/12)
    return sin, cos
```

In [66]:
```
data['sin_ARREST'] = circular_encoding('ARREST_DATE')[0]
data['cos_ARREST'] = circular_encoding('ARREST_DATE')[1]

data['sin_DISPOSITION'] = circular_encoding('DISPOSITION_DATE')[0]
data['cos_DISPOSITION'] = circular_encoding('DISPOSITION_DATE')[1]
```

- Keeping the Sentence Year only to represent the Case YEAR

Raw date columns are removed as they are no longer needed.

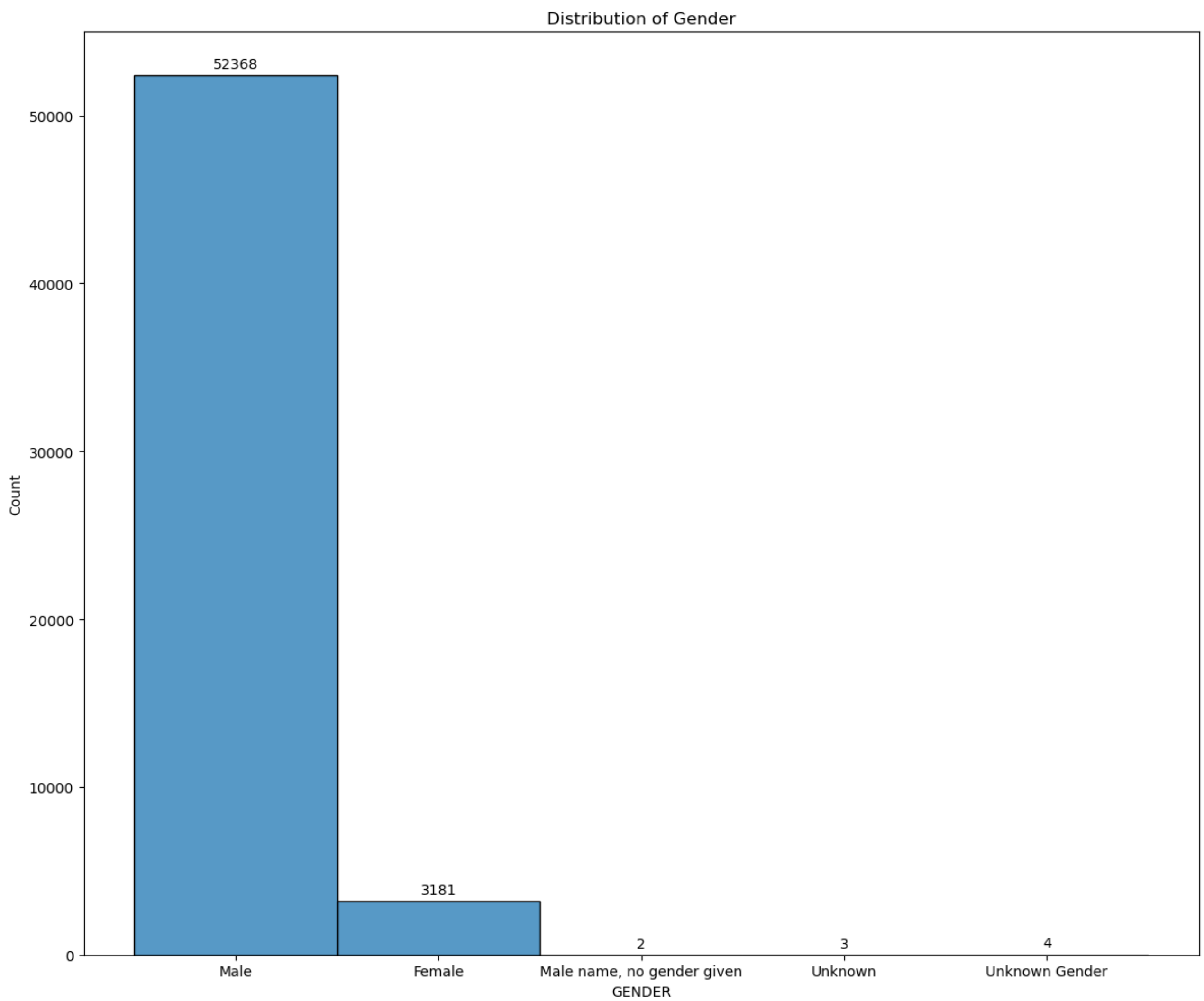
In [67]:
```
data['YEAR'] = data['SENTENCE_DATE'].dt.year
```

In [68]:
```
data = data.drop(date_col, axis = 1)
```

See EDA and Machine Learning notebook for further analysis regarding circular encoded data

Data Cleaning: Gender

In [69]:
```
plt.figure(figsize = (12,10))
plt.title('Distribution of Gender')
gender_plt = sns.histplot(data['GENDER'])
bar_label(gender_plt)
plt.tight_layout()
```



It seems that there are two different types of "Unknown." One gender is implied from a male name.

Drop non-male or female genders.

In [70]:
```python
data = data[data['GENDER'].isin(['Male','Female'])].reset_index(drop = True)
```

Data Cleaning: Race

In [71]:
```python
data['RACE'].value_counts()
```

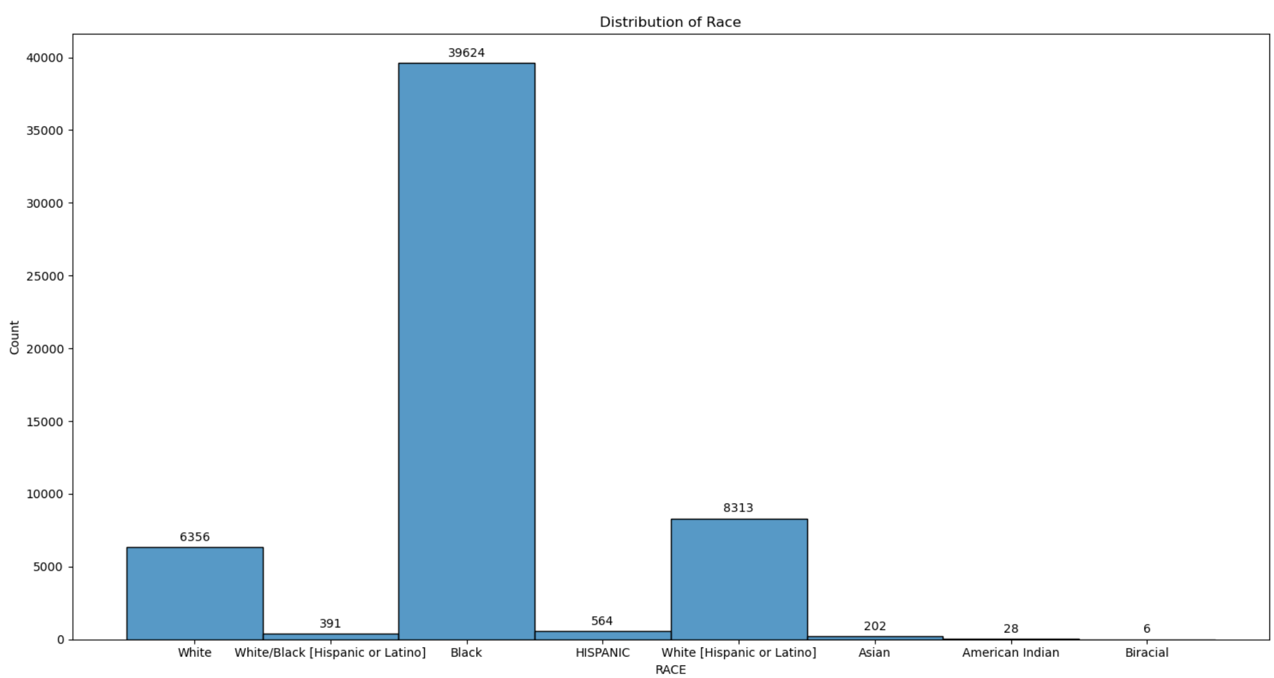
Out[71]:
```
Black                                 39624
White [Hispanic or Latino]             8313
White                                  6356
HISPANIC                                564
White/Black [Hispanic or Latino]        391
Asian                                   202
Unknown                                  65
American Indian                          28
Biracial                                  6
Name: RACE, dtype: int64
```

Unknown RACE will be dropped as it is not possible to impute.

Future iterations could consider further consolidation of the race groups.

In [72]:
```python
data = data[data['RACE']!='Unknown'].reset_index(drop = True)
```

In [73]:
```python
plt.figure(figsize = (15,8))
plt.title('Distribution of Race')
race_plt = sns.histplot(data['RACE'])
bar_label(race_plt)
plt.tight_layout()
```



Feature Engineering: Standardizing Commitment Term

- Commitment Term in Days
    - Turn all commitment terms into days using the commitment unit.
    - Drop the commitment unit once complete

In [74]:
```
data['COMMITMENT_UNIT'].value_counts()
```

Out[74]:
```
Year(s)          50305
Months            5098
Natural Life        42
Days                27
Term                 8
Dollars              3
Hours                1
Name: COMMITMENT_UNIT, dtype: int64
```

Units Natural Life, Term, Dollars, and Hours are unusual.

In [75]:
```
data[data['COMMITMENT_UNIT'].isin(['Natural Life','Term','Dollars','Hours'])]
.sort_values(by = 'COMMITMENT_UNIT')
```

Out[75]:

	OFFENSE_CATEGORY	CHARGE_COUNT	DISPOSITION_CHARGED_CLASS	SENTENCE_
20110	Possession of Stolen Motor Vehicle	1	2	Joan M
36582	Burglary	1	2	Allen F
7562	Driving With Suspended Or Revoked License	1	4	Colleen Ann
34085	UUW - Unlawful Use of Weapon	1	4	Alfredo Mal
516	Armed Robbery	1	X	Timothy
19618	Homicide	1	M	Colleen Ann
21639	Homicide	1	M	Joanne
24543	Homicide	1	M	Byrne, T
24544	Homicide	1	M	Byrne, T
24883	Aggravated Battery With A Firearm	1	X	Maura S
24949	Homicide	1	M	Byrne, T
24950	Homicide	1	M	Byrne, T

27589	Sex Crimes	1	X	Richard E S
19113	Sex Crimes	1	X	Timothy
29764	Homicide	1	M	Vincent M G
29901	Homicide	1	M	Timothy
35227	Homicide	1	M	Thaddeus L
39519	Homicide	1	M	Timothy
39520	Homicide	1	M	Timothy
41386	Homicide	1	M	W Gar
48082	Sex Crimes	1	X	Kenworthy
49742	UUW - Unlawful Use of Weapon	1	X	Dom Step
26327	Homicide	1	M	Nicholas
18328	Sex Crimes	1	X	Alfredo Mal
18353	Sex Crimes	1	X	Alfredo Mal
15838	Sex Crimes	1	X	Jo Kaz
2843	Other Offense	1	X	Timothy
1940	Sex Crimes	1	X	Kenneth J
3013	Homicide	1	M	Charles
3511	Other Offense	1	X	Dom Step
1482	Armed Robbery	1	X	James
16951	Sex Crimes	1	X	Allen F
5412	UUW - Unlawful Use of Weapon	1	X	Dennis
5816	Homicide	1	M	Charles

7224	Armed Robbery	1	X	Noreen Valer
7691	Homicide	1	M	Charles
9036	Attempt Armed Robbery	1	M	Stanley
9988	Armed Robbery	1	X	Erica L
10379	Homicide	1	M	Charles
11354	Homicide	1	M	Erica L
12168	Sex Crimes	1	X	Alfredo Mal
1162	Armed Robbery	1	X	Rosemar
15829	Sex Crimes	1	X	Jo Kaz
1134	Homicide	1	M	Dom Step
2825	Homicide	1	M	James
2088	Homicide	1	M	Mary M Bro
43552	Possession of Stolen Motor Vehicle	1	2	Micha
29004	UUW - Unlawful Use of Weapon	1	4	Neil J
4454	Retail Theft	1	4	Shelley
14286	Driving With Suspended Or Revoked License	1	4	Brian K
19497	Driving With Suspended Or Revoked License	1	4	Michele M
53723	Aggravated Battery Police Officer	1	2	Margaret
35213	UUW - Unlawful Use of Weapon	1	4	Neil J
55226	Retail Theft	1	3	Terry Ga

Any cases where the commitment unit is in terms of dollars or hours will be dropped. The unit "term" is ambiguous in this instance and impossible to impute. "Dollar" and "hour" appear to be errors or outliers, as a prison sentence typically lasts longer than hours.

Natural life sentences with a duration of 0 will also be dropped.

In [76]:
```
data = data[(~data['COMMITMENT_UNIT'].isin(['Term','Dollars','Hours']))]
```

In [77]:
```
data = data[~((data['COMMITMENT_UNIT']=='Natural Life')&(data['COMMITMENT_TE
```

Natural life sentences mean that the participant is sentenced to prison for the rest of their lives. In order to compute this back to a sentence in days, the 95th percentile sentence length in years for the same age and offense class will be considered to convert the natural life sentence into years. Another potential method for approximating a natural life sentence could be 105 - age.

In [78]:
```
age_class_df = pd.DataFrame(data=[list(idx) for idx in data[data['COMMITMENT
                                                                        'AGE_AT
                                  columns = ['DISPOSITION_CHARGED_CLASS','AGE_AT_IN
```

In [79]:
```
for x in range(len(age_class_df)):
    c_class = age_class_df.iloc[x,0]
    age = age_class_df.iloc[x,1]
    result = data[(data['DISPOSITION_CHARGED_CLASS']==c_class)&(data['AGE_AT
&(data['AGE_AT_INCIDENT']<=(age+2))&(data['COMMITMENT_UNIT']=='Year(s)')].co
    if len(result)>0:
        year = result['COMMITMENT_TERM'].quantile(.95) # obtain 95th percent
    else:
        year = np.nan
    age_class_df.loc[x,'Years'] = year
```

In [80]:
```
age_class_df
```

Out[80]:

	DISPOSITION_CHARGED_CLASS	AGE_AT_INCIDENT	Years
0	M	19.0	55.50
1	M	21.0	64.40
2	M	22.0	65.00
3	M	23.0	65.00
4	M	28.0	65.00
5	M	34.0	52.90
6	X	48.0	21.00
7	X	23.0	25.00
8	X	32.0	25.00
9	X	37.0	25.00
10	X	52.0	20.00
11	X	50.0	20.55
12	X	46.0	24.40
13	X	44.0	25.00
14	X	41.0	25.00
15	X	40.0	25.00
16	X	28.0	25.00
17	X	35.0	25.00
18	X	33.0	25.00
19	M	20.0	63.00
20	M	48.0	46.50
21	M	46.0	35.00
22	M	38.0	55.00
23	M	29.0	65.00
24	X	57.0	25.00

In [81]:

```
for x in range(len(age_class_df)):
    c_class = age_class_df.iloc[x,0]
    age = age_class_df.iloc[x,1]
    year = age_class_df.iloc[x,2]
    index = data[(data['DISPOSITION_CHARGED_CLASS']==c_class)&(data['AGE_AT_
&(data['AGE_AT_INCIDENT']<=(age+2))&(data['COMMITMENT_UNIT']=='Natural Life'
    data.loc[index,'COMMITMENT_UNIT'] = 'Year(s)'
    data.loc[index,'COMMITMENT_TERM'] = year
```

The following operations are performed on each commitment term/unit to convert the commitment term into days:

Years *365, Months* 30

The Commitment Term Units feature can be dropped after standardization.

In [82]:
```
data['COMMITMENT_TERM'] = np.where(data['COMMITMENT_UNIT']=='Year(s)',data['
          np.where(data['COMMITMENT_UNIT']=='Months',data['COMMITMENT_TERM']*3
```

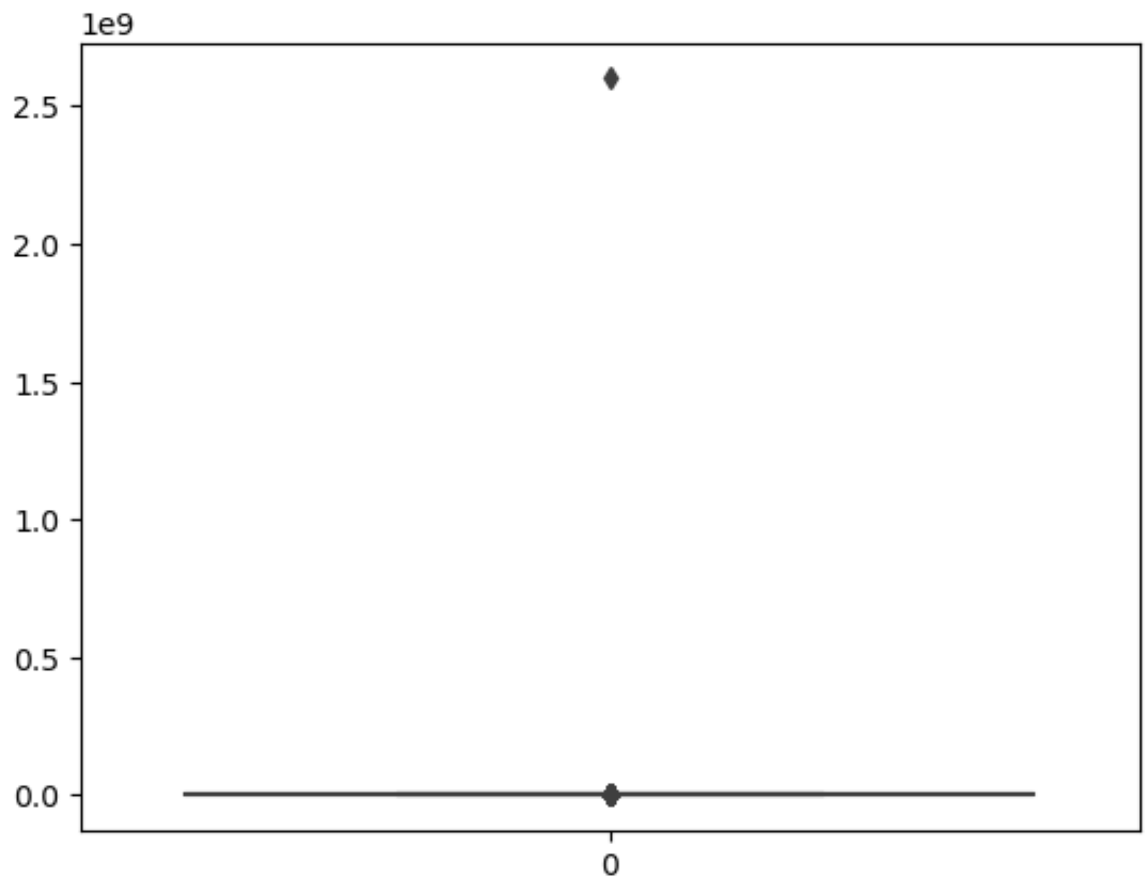
In [83]:
```
data = data.drop(['COMMITMENT_UNIT'], axis = 1)
```

Data Cleaning: Outlier Commitment Term

There appears to be a huge outlier commitment term:

In [84]:
```
sns.boxplot(data['COMMITMENT_TERM'])
```

Out[84]:
```
<Axes: >
```



In [85]:
```
data.loc[data['COMMITMENT_TERM'].idxmax(),:]
```

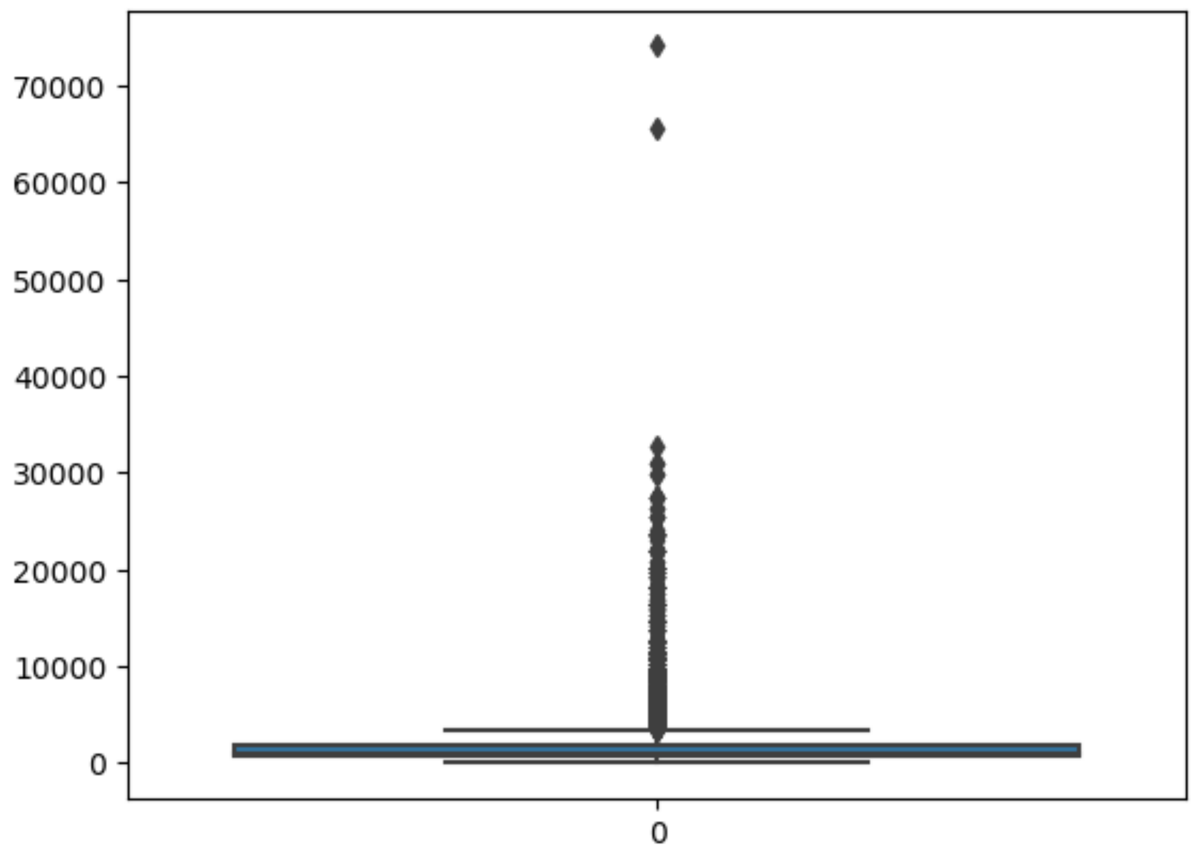
Out[85]:
```
OFFENSE_CATEGORY                                           UUW - Unlawful Use o
f Weapon
CHARGE_COUNT
1
DISPOSITION_CHARGED_CLASS
2
SENTENCE_JUDGE                                                     Geraldine A
D'Souza
SENTENCE_COURT_NAME                                               District 6 -
Markham
SENTENCE_COURT_FACILITY                                             Markham Co
urthouse
COMMITMENT_TERM                                                           2599
538395.0
AGE_AT_INCIDENT
23.0
RACE
Black
GENDER
Male
INCIDENT_CITY
Dolton
LAW_ENFORCEMENT_AGENCY       COOK COUNTY SHERIFF'S POLICE PATROL MARKHAM (IL
016B100)
arrest_incident
0
sentence_arrest
898
sin_ARREST
0.866025
cos_ARREST
0.5
sin_DISPOSITION                                                              -
0.866025
cos_DISPOSITION
-0.5
YEAR
2023
Name: 51263, dtype: object
```

Outlier term is dropped.

In [86]:
```
data = data.drop(51263)
```

In [87]:
```
sns.boxplot(data['COMMITMENT_TERM'])
```

Out[87]:
```
<Axes: >
```



If an extreme sentence is considered > 99 years, or 36,165 days, there still appears to be two outliers with over 60,000 days.

In [88]:
```
data[data['COMMITMENT_TERM']>40000][['DISPOSITION_CHARGED_CLASS','OFFENSE_CA
                                      'COMMITMENT_TERM']]
```

Out[88]:

	DISPOSITION_CHARGED_CLASS	OFFENSE_CATEGORY	AGE_AT_INCIDENT	COMMITM
45817	1	Homicide	18.0	
48632	4	Burglary	44.0	

Since the severity of the crimes between outliers seems drastically different (homicide vs burglary), it is highly unlikely that the two errors would have had similar sentences.

Comparing the sentence length of other young participants, the outlier may have been caused by an additional 0 at the end of the sentence. It seems plausible that the true sentence was 20.3 years or 7409.5 days. However, since the commitment term is the predicted value, we will be dropping the outlier due to uncertainty.

In [89]:
```
data[(data['OFFENSE_CATEGORY']=='Homicide')&(data['DISPOSITION_CHARGED_CLASS
    (data['AGE_AT_INCIDENT']<=20)]['COMMITMENT_TERM'].sort_values()
```

Out[89]:
```
11329     1460.0
50991     1642.5
18643     2555.0
13855     2555.0
10228     2555.0
           ...
33206     7300.0
32877     7300.0
14800     7300.0
19495     7300.0
45817    74095.0
Name: COMMITMENT_TERM, Length: 63, dtype: float64
```

Looking at other similar burglaries, it is unclear what the true commitment term should have been. The outlier is dropped due to the uncertainty.

In [90]:
```
data[(data['OFFENSE_CATEGORY']=='Burglary')&(data['DISPOSITION_CHARGED_CLASS
    (data['AGE_AT_INCIDENT']<=46)&(data['AGE_AT_INCIDENT']>=42)]['COMMITMENT
```

Out[90]:
```
989       365.0
54261     365.0
53399     365.0
50965     365.0
50504     365.0
50432     365.0
48816     365.0
39335     365.0
18962     365.0
55240     365.0
29625     730.0
55167     730.0
41959     730.0
12951     730.0
10351     730.0
35495    1095.0
40091    1095.0
41387    1095.0
24182    1095.0
10491    1620.0
34013    1825.0
14053    1825.0
10023    1825.0
48632   65700.0
Name: COMMITMENT_TERM, dtype: float64
```

In [91]:
```
data = data.drop([45817,48632]).reset_index(drop = True)
```

See EDA and Machine Learning notebook for further analysis regarding commitment time and other continuous features

Feature Engineering: Judge Names

As there are a lot of unique Judge Names, a simple encoding is not optimal. Instead, groups of judges will be created using the first Sentencing Year.

Prior to creating the group, however, Judge Names could be repeated but with different punctuations, spaces, etc.

In [92]:
```
from Levenshtein import distance as levenshtein_distance
judge_names = data['SENTENCE_JUDGE'].str.upper().replace(' ', '', regex=True

# Levenshtein distance calculates the number of charecter changes required t
def calculate_distance(judge1, judge2):
    score = levenshtein_distance(judge1, judge2)
    score = score / max(len(judge1),len(judge2)) # score is created by divic
    return score

result_df = pd.DataFrame(columns = ['Judge','Matching_Judge','Score'])
for current_judge in judge_names:
    other_judges = [x for x in judge_names if x != current_judge] # compare
    best_score = 1
    for other_judge in other_judges:
        score = calculate_distance(current_judge, other_judge)
        if score < best_score:
            best_score = score # find the best score and matching judge name
            best_match = other_judge
    result_df.loc[len(result_df),:] = [current_judge, best_match, best_score
```

In [93]:
```
result_df.sort_values(by = 'Score', ascending = True)[:10]
```

Out[93]:

	Judge	Matching_Judge	Score
143	WILLIAMBRAINES	WILLIAMRAINES	0.071429
86	WILLIAMRAINES	WILLIAMBRAINES	0.071429
119	DONALDRHAVIS	RONALDSDAVIS	0.25
135	RONALDSDAVIS	DONALDRHAVIS	0.25
122	SUSANSULLIVAN	LAURAMSULLIVAN	0.285714
81	LAURAMSULLIVAN	SUSANSULLIVAN	0.285714
123	MICHAELKANE	MICHAELCLANCY	0.307692
89	MICHAELBHYMAN	MICHAELBROWN	0.307692
164	MICHAELCLANCY	MICHAELKANE	0.307692
64	MICHAELBROWN	MICHAELBHYMAN	0.307692

William B Raines and William Raines could be the same judge.

In [94]:
```
data[(data['SENTENCE_JUDGE'].str.contains('William', case = False, na = Fals
(data['SENTENCE_JUDGE'].str.contains('Raines', case = False, na = False))][[
                                                                             '
```

Out[94]:
```
YEAR  SENTENCE_COURT_NAME      SENTENCE_COURT_FACILITY
2018  District 5 - Bridgeview  Bridgeview Courthouse      89
2019  District 1 - Chicago     26TH Street                80
2017  District 5 - Bridgeview  Bridgeview Courthouse      75
2021  District 1 - Chicago     26TH Street                56
2020  District 1 - Chicago     26TH Street                20
2022  District 1 - Chicago     26TH Street                13
2018  District 1 - Chicago     26TH Street                 3
2016  District 5 - Bridgeview  Bridgeview Courthouse       1
2023  District 1 - Chicago     26TH Street                 1
dtype: int64
```

Upon looking into William Raines, it appears that William B Raines is the only judge in the Cook County Judicial Circuit.

The judge name will be standardized to William Raines.

In [95]:
```
data['SENTENCE_JUDGE'] = np.where((data['SENTENCE_JUDGE'].str.contains('Will
(data['SENTENCE_JUDGE'].str.contains('Raines', case = False, na = False)),'W
```

- Groups of Judges by first Sentence Year
  - The data contains cases from 2011 to present.
  - Judges will be split into four groups using quantiles.

Drop judge name once complete.

In [96]:
```
judge_yrs = pd.DataFrame(data.groupby('SENTENCE_JUDGE')['YEAR'].min().copy()
```

In [97]:
```
judge_yrs['Judge_Group'] = np.where(judge_yrs['YEAR']<=judge_yrs['YEAR'].qua
         np.where(judge_yrs['YEAR']<=judge_yrs['YEAR'].quantile(.5),'Moderat
                  np.where(judge_yrs['YEAR']<=judge_yrs['YEAR'].quantile(.75
```

In [98]:
```
data = pd.merge(data,judge_yrs[['Judge_Group','SENTENCE_JUDGE']], how = 'inn
```

In [99]:
```
data = data.drop('SENTENCE_JUDGE', axis = 1)
```

Feature Engineering: City

- Creating a Chicago vs Non-Chicago Bucket
  - As there are over 100 unique cities in the data set, a binary group of cities will be created. Chicago is the largest city within Cook County and accounts for the most cases in the data set. A simple Incident in Chicago or Not in Chicago will be created.
  - While Sentence Court Name / Court Facility provides location information, the sentencing may not always be in the same city as the incident.

Dropping City once complete.

In [100…
```
data[['INCIDENT_CITY','SENTENCE_COURT_NAME']].value_counts()
```

Out[100]:
```
INCIDENT_CITY      SENTENCE_COURT_NAME
Chicago            District 1 - Chicago            29744
                   District 2 - Skokie              6526
                   District 5 - Bridgeview          3617
Cicero             District 4 - Maywood              631
Harvey             District 6 - Markham              478
                                                    ...
Hometown           District 1 - Chicago                1
                   District 6 - Markham                1
Olympia FIELDS     District 6 - Markham                1
Oakbrook Terrace   District 2 - Skokie                 1
Elmwood Park       District 3 - Rolling Meadows        1
Length: 315, dtype: int64
```

In [101…
```
data['Incident_Chicago'] = np.where(data['INCIDENT_CITY']=='Chicago',1,0)

data = data.drop('INCIDENT_CITY', axis = 1)
```

Data Cleaning: Droping Final Unused Columns

- Law Enforcement Agency:
  - While Law Enforcement Agency was originally kept instead of Law Enforcement Unit, the Incident_Chicago flag contains similar information regarding location. The Law Enforcement Agency would have likely been an agency from the same city as the incident.
- Sentence Court Facility
  - The Sentence Court Name contains most of the information the Facility Name would have provided. Chicago contains 7 facilities, with most cases held at the 26th Street facility. All other court names only have one court facility.

In [102…
```
data.groupby('SENTENCE_COURT_NAME')['SENTENCE_COURT_FACILITY'].value_counts(
```

Out[102]:
```
SENTENCE_COURT_NAME           SENTENCE_COURT_FACILITY
District 1 - Chicago          26TH Street                          30062
                              DV Courthouse                           37
                              Harrison & Kedzie (Area 4)              10
                              727 E. 111th Street (Area 2)             3
                              Belmont & Western (Area 3)               3
                              51st & Wentworth (Area 1)                2
                              Grand & Central (Area 5)                 1
District 2 - Skokie           Skokie Courthouse                     8235
District 3 - Rolling Meadows  Rolling Meadows Courthouse            2453
District 4 - Maywood          Maywood Courthouse                    3217
District 5 - Bridgeview       Bridgeview Courthouse                 6729
District 6 - Markham          Markham Courthouse                    4711
Name: SENTENCE_COURT_FACILITY, dtype: int64
```

In [103…
```
data = data.drop(['LAW_ENFORCEMENT_AGENCY','SENTENCE_COURT_FACILITY'], axis
```

Feature Engineering Dummy Variables

- Gender, Incident Chicago, Judge Group, Sentence Court Name, Race, Disposition Charged Class, Offense Category
    - Incident Chicago is already one hot encoded.

The above categorical variables will be encoded into dummy variables, dropping the first for k-1 categories.

- See EDA and Machine Learning notebook for further analysis regarding categorical data

In [104…
```
data.nunique().sort_values()
```

Out[104]:
```
Incident_Chicago                2
GENDER                          2
Judge_Group                     4
SENTENCE_COURT_NAME             6
RACE                            8
sin_ARREST                     11
sin_DISPOSITION                11
DISPOSITION_CHARGED_CLASS      12
cos_DISPOSITION                12
cos_ARREST                     12
YEAR                           15
CHARGE_COUNT                   27
AGE_AT_INCIDENT                64
OFFENSE_CATEGORY               84
COMMITMENT_TERM               207
arrest_incident              1356
sentence_arrest              1967
dtype: int64
```

In [105…
```
data.isnull().sum()
```

Out[105]:
```
OFFENSE_CATEGORY              0
CHARGE_COUNT                  0
DISPOSITION_CHARGED_CLASS     0
SENTENCE_COURT_NAME           0
COMMITMENT_TERM               0
AGE_AT_INCIDENT               0
RACE                          0
GENDER                        0
arrest_incident               0
sentence_arrest               0
sin_ARREST                    0
cos_ARREST                    0
sin_DISPOSITION               0
cos_DISPOSITION               0
YEAR                          0
Judge_Group                   0
Incident_Chicago              0
dtype: int64
```

In [106…
```
data_dummy = pd.get_dummies(data, columns = ['GENDER','OFFENSE_CATEGORY','DI
                                  'RACE','Judge_Group'], drop_first=True).reset_
# drop first to remove the redundant column
```

Export Data to Pickle for EDA

In [107…
```
data_dummy.to_pickle('data_dummy.pkl')
data.to_pickle('data.pkl')
```

Summary: Data Cleaning

In [108…
```
# No Dummy Variable
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 55463 entries, 0 to 55462
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   OFFENSE_CATEGORY            55463 non-null  object
 1   CHARGE_COUNT                55463 non-null  int64
 2   DISPOSITION_CHARGED_CLASS   55463 non-null  object
 3   SENTENCE_COURT_NAME         55463 non-null  object
 4   COMMITMENT_TERM             55463 non-null  float64
 5   AGE_AT_INCIDENT             55463 non-null  float64
 6   RACE                        55463 non-null  object
 7   GENDER                      55463 non-null  object
 8   arrest_incident             55463 non-null  int64
 9   sentence_arrest             55463 non-null  int64
 10  sin_ARREST                  55463 non-null  float64
 11  cos_ARREST                  55463 non-null  float64
 12  sin_DISPOSITION             55463 non-null  float64
 13  cos_DISPOSITION             55463 non-null  float64
 14  YEAR                        55463 non-null  int64
 15  Judge_Group                 55463 non-null  object
 16  Incident_Chicago            55463 non-null  int64
dtypes: float64(6), int64(5), object(6)
memory usage: 7.6+ MB
```

In [109…

```
# with Dummy Variables
data_dummy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55463 entries, 0 to 55462
Columns: 121 entries, CHARGE_COUNT to Judge_Group_Oldest
dtypes: float64(6), int64(5), uint8(110)
memory usage: 10.5 MB
```

Summary: Data Cleaning

After data cleaning, the dataset has the following dimensions:

- The cleaned pickle file is 12MB.
- The cleaned dataset contains 121 columns and 55,463 rows.
- There were 17 features prior to dummy variable encoding:
    - 7 features were categorical:
        - Gender, Offense Category, Disposition Charged Class, Sentence Court Name, Race, Judge Group, Incident Chicago.
    - 10 features were continuous variables, including the independent variable COMMITMENT_TERM:
        - Age, Charge Count, Year, Cosine and Sine of Disposition Month, Cosine and Sine of Arrest Month, Days between Sentence and Arrest Date, Days between Arrest and Incident Date, Commitment Term.

Outlier years were identified by first checking for any years greater than 2024. Outlier dates include cases where the receive date is after the arrest date, where the arrest occurs prior to the arraignment, where sentences occur prior to the arraignment, and where the incident date is after the arrest. All outlier years were corrected if the date or year was not ambiguous, and all outlier dates were removed if not imputable.

24 total features were dropped due to unrelated scope, containing a vast number of NAs, or were confirmed to have similar other features that provide similar information.

The month of the arrest and disposition were circular encoded using cosine and sine encoding. All categorical variables were turned into dummy variables, removing k-1 features to ensure there were no columns that did not contribute information.

Please refer to further feature engineering within the EDA and Machine Learning notebook. Some potential future difficulties lie in the large number of unique values within some categorical variables. Although extensive data cleaning was performed to rectify errors, fill missing values, treat outliers, and drop features providing redundant or irrelevant information, there are still columns with a high number of unique values. This may pose a challenge in both statistical modeling and machine learning applications due to the weakness in some of the features that were turned into dummy variables.

In [ ]:
```

```

In [28]:
```
# imports
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
import datetime

import statsmodels.formula.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

from statsmodels.tsa.stattools import adfuller

from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, cross_val_score,learni
from xgboost.sklearn import XGBRegressor
```

Import Data from Data Cleaning Steps

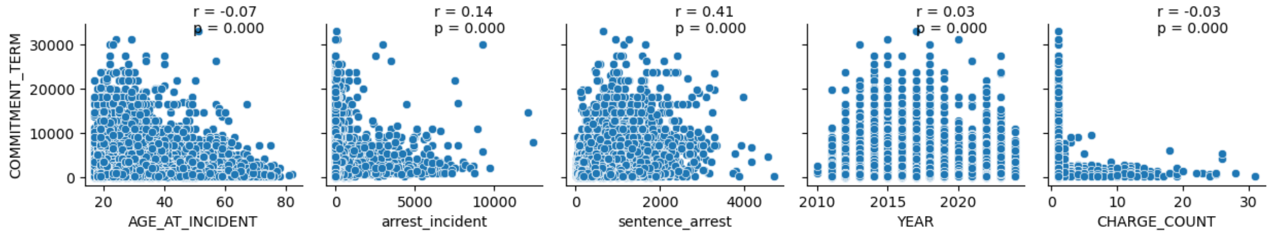
In [2]:
```
data = pd.read_pickle('data.pkl') # no dummy variables
data_dummy = pd.read_pickle('data_dummy.pkl') # with dummy variables
```

EDA: Pairplot with continuous variables

Since the sine and cosine circular encoded date variables do not make sense in a pairplot or correlation heatplot, they will be separately analyzed.

In [150…
```
def corrfunc(x, y, **kws):
    r, p = st.pearsonr(x, y) # plot the R2 and P value between two variables
    ax = plt.gca()
    ax.annotate("r = {:.2f}".format(r), xy=(0.5, 1.05), xycoords=ax.transAxe
    ax.annotate("p = {:.3f}".format(p), xy=(0.5, 0.95), xycoords=ax.transAxe

cont_var = ['AGE_AT_INCIDENT','arrest_incident','sentence_arrest','YEAR','CH
```

In [151…
```
pplot = sns.pairplot(data,
             y_vars = 'COMMITMENT_TERM',
            x_vars = cont_var)
pplot.map(corrfunc)
plt.tight_layout()
```



It appears that all five continuous variables have a linear relationship with Commitment Term. All five variables have a slope in the linear function that is not equal to 0.

One hypothesis from the Data Cleaning steps was that there could be a positive relationship between the time between Sentencing and Arrest, as it may indicate a complex case. Days between Sentence and Arrest dates seem to have a moderate positive relationship with Commitment Term.

Interestingly, charge count seems to have a negative relationship (albeit very weak) with Commitment Term. I believe this is mainly due to the large volume of cases with only one charge. Similarly, the year has a very weak positive relationship with Commitment Term.

The days between arrest and incident has a weak positive relationship with Commitment Term. Finally, the age of the participant also appears to have a weak to very weak negative relationship with Commitment Term.

In [152…

```
applot = sns.pairplot(data[['COMMITMENT_TERM']+cont_var])
applot.map(corrfunc)
plt.tight_layout()
```

The only potential collinear variable within the continuous variables appears to be YEAR and the arrest-sentence date. There appears to be a moderate positive linear relationship between the two variables. Because the R-squared of the two variables is less than 0.7, the risk for collinearity or multicollinearity appears to be low. Further testing across all variables will still be performed during model selection.

EDA: Temporal Relationship between Term length and Arrest and Disposition Dates

In [153…

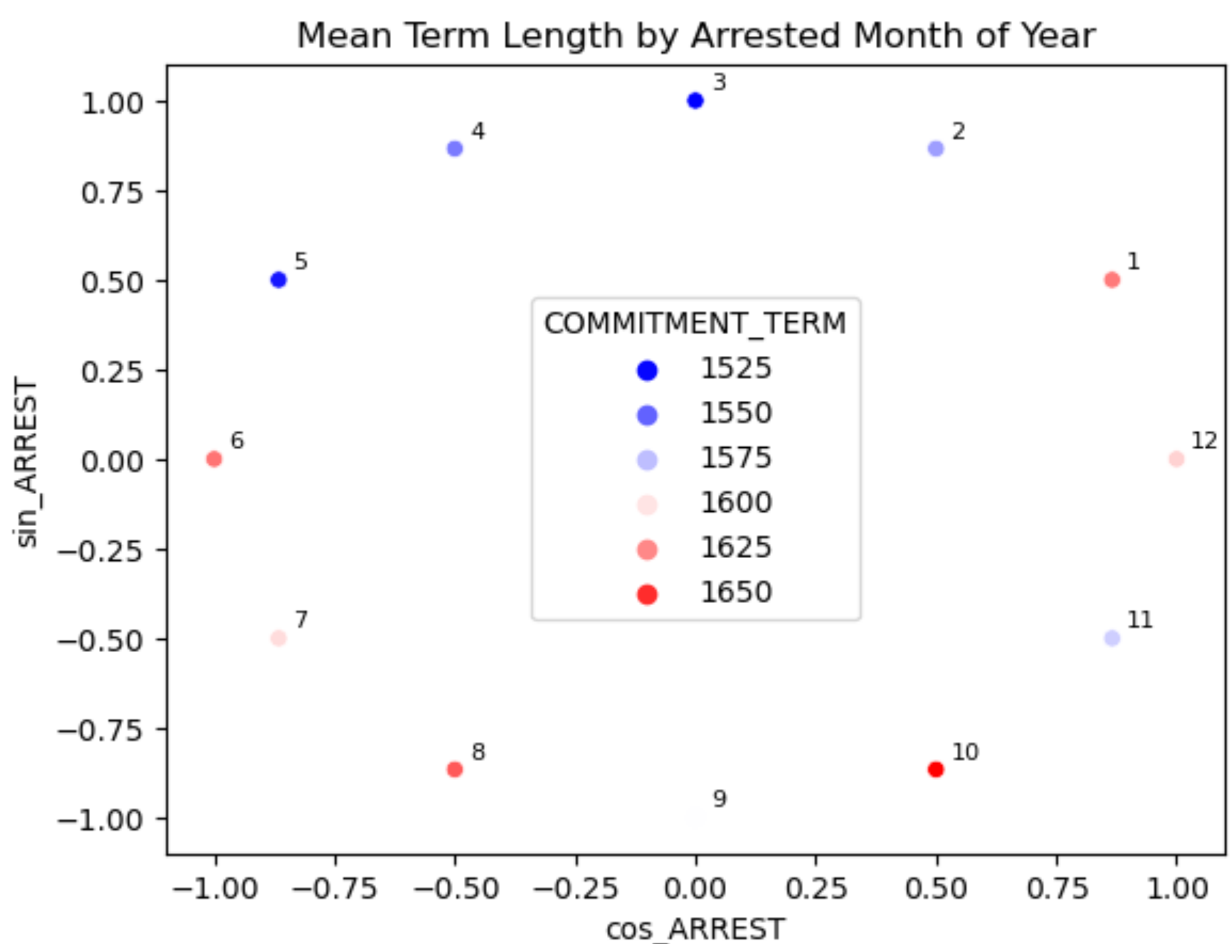
```
time_col = data.columns[(data.columns.str.contains('sin_'))|(data.columns.st
```

In [155…

```
# mean term length per Arrest month of year
arrest_time = data.groupby([time_col[0],time_col[1]])['COMMITMENT_TERM'].mea
# mean term length per disposition month of year
disposition_time = data.groupby([time_col[2],time_col[3]])['COMMITMENT_TERM'

# Decoding Sine and Cosine month of year
for x in range(1,13):
    sin = np.sin(2 * np.pi * x/12)
    cos = np.cos(2 * np.pi * x/12)
    arrest_ind = arrest_time[(arrest_time['cos_ARREST']==cos)&(arrest_time['
    disp_ind = disposition_time[(disposition_time['cos_DISPOSITION']==cos)&(
    arrest_time.loc[arrest_ind,'Month'] = x
    disposition_time.loc[disp_ind,'Month']=x
```

In [156…

```
plt.title('Mean Term Length by Arrested Month of Year')
sns.scatterplot(data=arrest_time, x='cos_ARREST', y='sin_ARREST',
                hue='COMMITMENT_TERM', palette='bwr')

for i in range(len(arrest_time)):
    plt.text(arrest_time['cos_ARREST'][i]+.03, arrest_time['sin_ARREST'][i]+
             str(int(arrest_time['Month'][i])),
            fontsize=8)
```



Linear Modeling Month Arrested and Term Length

In [390…
```
sm.ols(formula='COMMITMENT_TERM ~ sin_ARREST + cos_ARREST',
                data=data).fit().summary()
```

Out[390]:

OLS Regression Results

Dep. Variable:	COMMITMENT_TERM	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	6.583
Date:	Thu, 06 Jun 2024	**Prob (F-statistic):**	0.00138
Time:	20:57:31	**Log-Likelihood:**	-4.9647e+05
No. Observations:	55463	**AIC:**	9.929e+05
Df Residuals:	55460	**BIC:**	9.930e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1592.8629	7.946	200.455	0.000	1577.288	1608.438
sin_ARREST	-39.5313	11.235	-3.518	0.000	-61.552	-17.510
cos_ARREST	9.9391	11.218	0.886	0.376	-12.049	31.927

Omnibus:	55737.647	**Durbin-Watson:**	1.653
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3887343.122
Skew:	4.944	**Prob(JB):**	0.00
Kurtosis:	42.804	**Cond. No.**	1.42

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

There doesn't appear to be a linear relationship nor a seasonal trend between the arrested month and mean term length.

Arrests from February to May do seem to have a slightly lower mean term length. A t-test will be used since the population standard deviation is unknown.

Null Hypothesis: The mean sentence lengths for arrests between February and May are not different from the mean sentence lengths for arrests outside of February to May.

In [158…

```
def ttest(df1,df2):
    print('Mean 1: '+str(df1.mean()))
    print('Mean 2: '+str(df2.mean()))
    print(st.ttest_ind(df1,df2))
```

In [159…

```
in_group = arrest_time[(arrest_time['Month']>=2)&(arrest_time['Month']<=5)].
ttest(data[data.apply(lambda row: (row['sin_ARREST'], row['cos_ARREST']) in
     data[~data.apply(lambda row: (row['sin_ARREST'], row['cos_ARREST']) in
# Group 1: In Group (Feb-May Arrests)
# Group 2: Out Group
```

```
Mean 1: 1542.7144823324531
Mean 2: 1617.8080425893475
TtestResult(statistic=-4.420679102996416, pvalue=9.857773297526554e-06, df=5
5461.0)
```

In [160…

```
sm.ols(formula='COMMITMENT_TERM ~ Arrest_Flag',
                data=pd.concat([pd.Series(np.where(data.apply(lambda row: (r
         1,0)),data['COMMITMENT_TERM'].copy()],axis = 1).rename({0:'Arrest_F
```

Out[160]:

OLS Regression Results

Dep. Variable:	COMMITMENT_TERM	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	19.54
Date:	Tue, 04 Jun 2024	**Prob (F-statistic):**	9.86e-06
Time:	21:38:24	**Log-Likelihood:**	-4.9647e+05
No. Observations:	55463	**AIC:**	9.929e+05
Df Residuals:	55461	**BIC:**	9.930e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1617.8080	9.624	168.110	0.000	1598.946	1636.670
Arrest_Flag	-75.0936	16.987	-4.421	0.000	-108.388	-41.799

Omnibus:	55736.746	**Durbin-Watson:**	1.653
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3887587.542
Skew:	4.943	**Prob(JB):**	0.00
Kurtosis:	42.806	**Cond. No.**	2.42

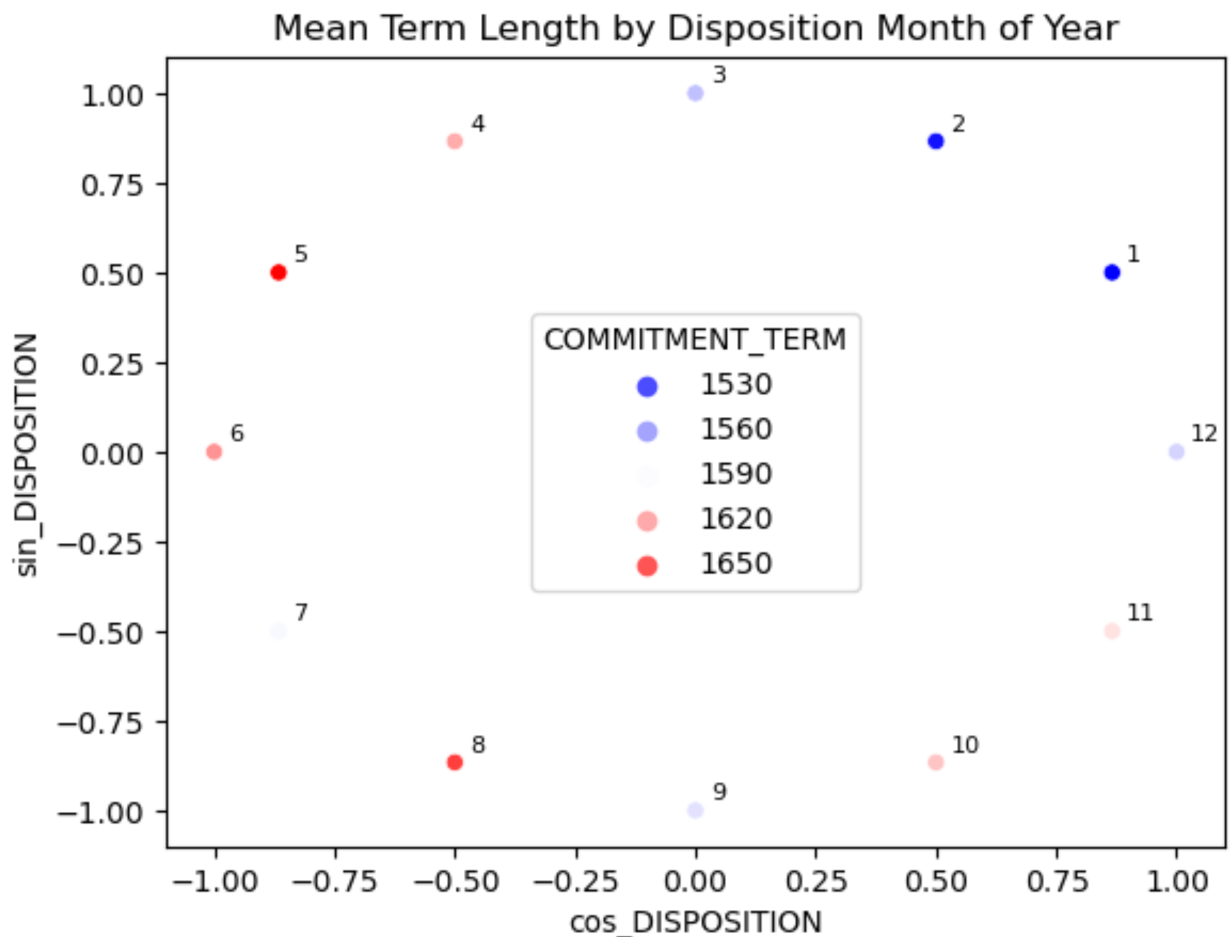
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Conclusion:

There appears to be a statistically significant difference between mean sentence lengths for arrests in February to May and those outside of February to May. However, the linear model using the in February to May flag and term length still showed no linear relationship. Therefore, this difference is likely due to random noise in the data or possibly due to the increase in crime volume during warmer months.

In [161…

```
plt.title('Mean Term Length by Disposition Month of Year')
sns.scatterplot(data=disposition_time, x='cos_DISPOSITION', y='sin_DISPOSITI
                hue='COMMITMENT_TERM', palette='bwr')

for i in range(len(disposition_time)):
    plt.text(disposition_time['cos_DISPOSITION'][i]+.03, disposition_time['s
             str(int(disposition_time['Month'][i])),
            fontsize=8)
```



In [162…

```
sm.ols(formula='COMMITMENT_TERM ~ sin_DISPOSITION + cos_DISPOSITION',
                data=data).fit().summary()
```

Out[162]:

OLS Regression Results

Dep. Variable:	COMMITMENT_TERM	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	10.72
Date:	Tue, 04 Jun 2024	**Prob (F-statistic):**	2.22e-05
Time:	21:38:25	**Log-Likelihood:**	-4.9647e+05
No. Observations:	55463	**AIC:**	9.929e+05
Df Residuals:	55460	**BIC:**	9.930e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1593.2763	7.931	200.896	0.000	1577.732	1608.821
sin_DISPOSITION	-23.2790	11.175	-2.083	0.037	-45.182	-1.376
cos_DISPOSITION	-46.1641	11.259	-4.100	0.000	-68.233	-24.095

Omnibus:	55734.496	**Durbin-Watson:**	1.652
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3889595.647
Skew:	4.943	**Prob(JB):**	0.00
Kurtosis:	42.817	**Cond. No.**	1.43

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

As with the mean term length by arrest month, the disposition month does not appear to have a linear relationship or a strong seasonal impact on the term length.

January and February do appear to have a lower mean term length. A t-test will be used since the population standard deviation is unknown.

Null Hypothesis: The mean sentence lengths for dispositions in January-February are not different from the mean sentence lengths for dispositions not in January-February.

In [163…

```
in_group = disposition_time[(disposition_time['Month']>=1)&(disposition_time
ttest(data[data.apply(lambda row: (row['sin_DISPOSITION'], row['cos_DISPOSIT
     data[~data.apply(lambda row: (row['sin_DISPOSITION'], row['cos_DISPOSIT
# Group 1: In Group (Jan-Feb Dispositions)
# Group 2: Out Group
```

```
Mean 1: 1506.828355113047
Mean 2: 1611.9881277143668
TtestResult(statistic=-5.025742797760597, pvalue=5.030442300578612e-07, df=5
5461.0)
```

In [164…

```
sm.ols(formula='COMMITMENT_TERM ~ Disposition_Flag',
                data=pd.concat([pd.Series(np.where(data.apply(lambda row: (r
        1,0)),data['COMMITMENT_TERM'].copy()],axis = 1).rename({0:'Disposit
```

Out[164]:

OLS Regression Results

Dep. Variable:	COMMITMENT_TERM	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	25.26
Date:	Tue, 04 Jun 2024	**Prob (F-statistic):**	5.03e-07
Time:	21:38:28	**Log-Likelihood:**	-4.9647e+05
No. Observations:	55463	**AIC:**	9.929e+05
Df Residuals:	55461	**BIC:**	9.930e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1611.9881	8.724	184.770	0.000	1594.888	1629.088
Disposition_Flag	-105.1598	20.924	-5.026	0.000	-146.171	-64.148

Omnibus:	55732.154	**Durbin-Watson:**	1.652
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3888099.106
Skew:	4.943	**Prob(JB):**	0.00
Kurtosis:	42.809	**Cond. No.**	2.73

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Conclusion:

- There appears to be a statistically significant difference between mean sentence lengths for dispositions in Jan-Feb and those outside of Jan-Feb.
- However, the linear model using the in Jan-Feb flag and term length still showed no linear relationship.
- Therefore, this difference is likely due to random noise in the data, but still possibly related to the holiday seasons (post-Christmas/New Years).

EDA: Distribution of Term length by Categorical Variables

In [313…
```
cat_var = data.columns[(~data.columns.isin(cont_var+time_col.tolist()))&(dat
```

In [185…
```
print(cat_var)
data[cat_var].nunique()
```

```
Index(['OFFENSE_CATEGORY', 'DISPOSITION_CHARGED_CLASS', 'SENTENCE_COURT_NAM
E',
       'RACE', 'GENDER', 'Judge_Group', 'Incident_Chicago'],
      dtype='object')
```

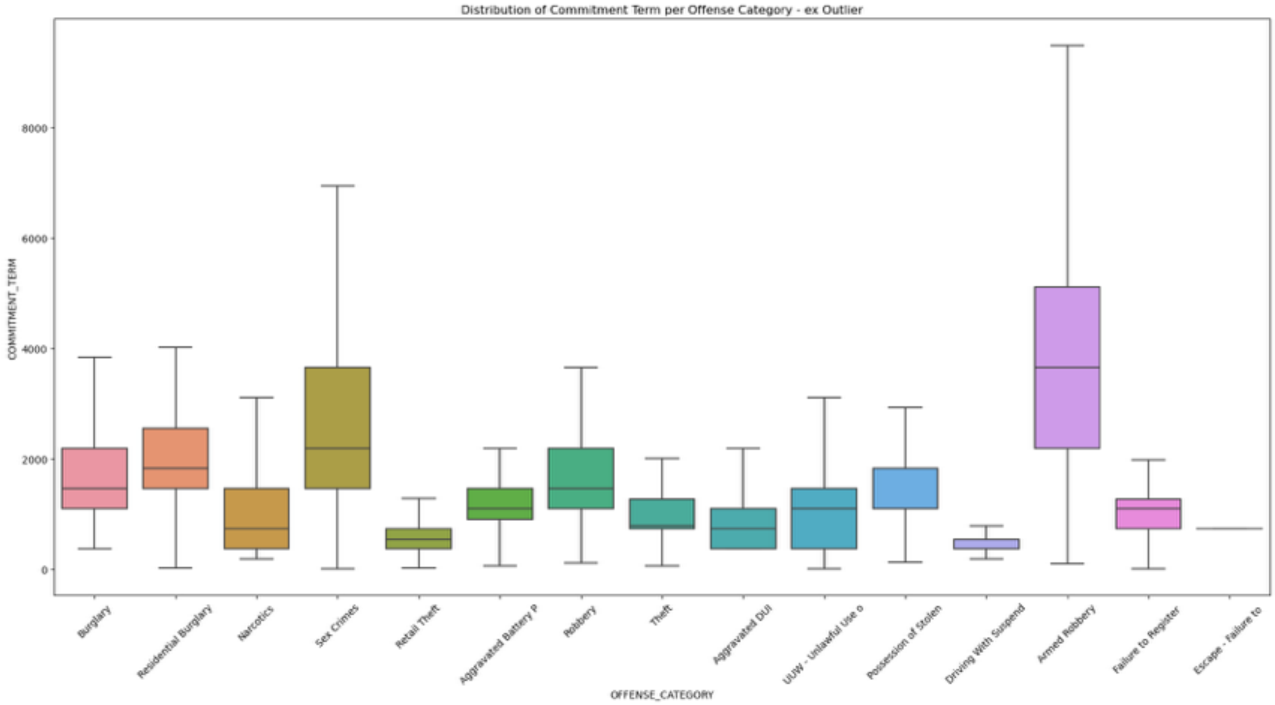
Out[185]:
```
OFFENSE_CATEGORY             84
DISPOSITION_CHARGED_CLASS    12
SENTENCE_COURT_NAME           6
RACE                          8
GENDER                        2
Judge_Group                   4
Incident_Chicago              2
dtype: int64
```

Offense Category

Since Offense Category has 84 unique variables, it will be analyzed separately. Due to the vast number of unique variables, the top 15 categories by frequency will be studied further.

In [194…
```
# only the top 15 categories are kept for the box plot
o_cat_15 = data[data[cat_var[0]].isin(data[cat_var[0]].value_counts().head(1
```

In [336…
```
plt.figure(figsize = (18,10))
plt.title('Distribution of Commitment Term per Offense Category - ex Outlier
bbplot = sns.boxplot(o_cat_15, x = 'OFFENSE_CATEGORY', y = 'COMMITMENT_TERM'
labels = o_cat_15['OFFENSE_CATEGORY'].unique()
bbplot.set_xticklabels([label[:20] for label in labels], rotation = 45) # tr
plt.tight_layout()
```



It appears that most median Commitment Term categories are around 1,500 days. However, Armed Robbery does appear to have a significantly higher median.

An ANOVA test will be conducted to determine if there are at least two or more groups with different means.

Null Hypothesis: There are no differences between all means.

In [331…

```
def print_means(df, col, category):
    for x in range(len(category)):
        print(str(category[x])+': '+str((df[df[col]==category[x]]['COMMITMEN
```

In [347…

```
category = o_cat_15['OFFENSE_CATEGORY'].unique()
print_means(o_cat_15,'OFFENSE_CATEGORY',category)
st.f_oneway(o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[0]]['COMMITMENT_
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[1]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[2]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[3]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[4]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[5]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[6]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[7]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[8]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[9]]['COMMITMENT_T
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[10]]['COMMITMENT_
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[11]]['COMMITMENT_
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[12]]['COMMITMENT_
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[13]]['COMMITMENT_
           o_cat_15[o_cat_15['OFFENSE_CATEGORY']==category[14]]['COMMITMENT_
```

```
Burglary: 1762.2185920239656
Residential Burglary: 2120.1060732113147
Narcotics: 1177.215786637931
Sex Crimes: 2941.5623337765956
Retail Theft: 625.8365467009426
Aggravated Battery Police Officer: 1253.9081261370527
Robbery: 1803.0038674033149
Theft: 1065.171986970684
Aggravated DUI: 874.9175779667881
UUW - Unlawful Use of Weapon: 1144.9337660587548
Possession of Stolen Motor Vehicle: 1470.2604294478529
Driving With Suspended Or Revoked License: 505.2258541392904
Armed Robbery: 3996.867484450587
Failure to Register as a Sex Offender: 1142.0400161681487
Escape - Failure to Return: 808.2325487012987
```

Out[347]:

```
F_onewayResult(statistic=2015.1437921317165, pvalue=0.0)
```

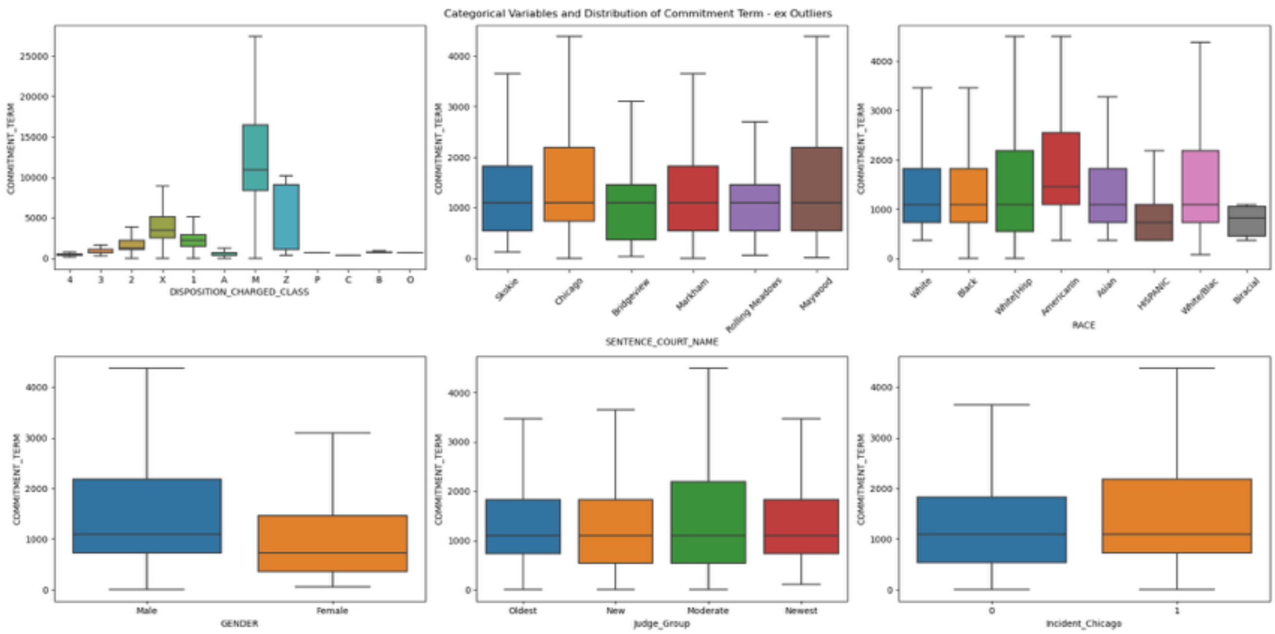
Conclusion:

Since the p-value is 0, there is a statistically significant difference in the mean Commitment Term for at least two of the top 15 offense categories. We will proceed to fit the model with the dummy variable of Offense Category.

EDA - Categorical Variables Continued

In [312…

```
fig,ax = plt.subplots(nrows = 2, ncols =3, figsize=(20, 10))
plt.suptitle('Categorical Variables and Distribution of Commitment Term - ex
class_plt = sns.boxplot(data, y = 'COMMITMENT_TERM', x = 'DISPOSITION_CHARGE
labels = data['DISPOSITION_CHARGED_CLASS'].unique()
class_plt.set_xticklabels([label[:15] for label in labels]) # trim label len

court_plt = sns.boxplot(data, y = 'COMMITMENT_TERM', x = 'SENTENCE_COURT_NAM
labels = data['SENTENCE_COURT_NAME'].unique()
court_plt.set_xticklabels([label[13:30] for label in labels], rotation=45)

race_plt = sns.boxplot(data, y = 'COMMITMENT_TERM', x = 'RACE', showfliers =
labels = data['RACE'].unique()
race_plt.set_xticklabels([label.replace(' ','')[:10] for label in labels], r

gender_plt = sns.boxplot(data, y = 'COMMITMENT_TERM', x = 'GENDER', showflie
judge_plt = sns.boxplot(data, y = 'COMMITMENT_TERM', x = 'Judge_Group', show
chi_plt = sns.boxplot(data, y = 'COMMITMENT_TERM', x = 'Incident_Chicago', s
plt.tight_layout()
```



Charged Class

The disposition charged class appears to significantly impact the commitment term. "M": first-degree murder appears to have the largest impact on sentencing length.

An ANOVA test will be conducted to determine if at least one non-"M" charge class has a statistical difference in sentencing term.

Null Hypothesis: There are no differences between all means.

In [348…

```
category = data['DISPOSITION_CHARGED_CLASS'].unique()
print_means(data,'DISPOSITION_CHARGED_CLASS',category)
st.f_oneway(data[data['DISPOSITION_CHARGED_CLASS']==category[0]]['COMMITMENT
           data[data['DISPOSITION_CHARGED_CLASS']==category[1]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[2]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[3]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[4]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[5]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[7]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[8]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[9]]['COMMITMENT_
           data[data['DISPOSITION_CHARGED_CLASS']==category[10]]['COMMITMENT
           data[data['DISPOSITION_CHARGED_CLASS']==category[11]]['COMMITMENT
```

```
4: 514.4362620469461
3: 985.2800594638178
2: 1608.700511322133
X: 4180.4469669615455
1: 2426.9786135693216
A: 515.7685185185185
M: 12540.135416666666
Z: 4048.181818181818
P: 730.0
C: 365.0
B: 786.6666666666666
O: 730.0
```

Out[348]:
```
F_onewayResult(statistic=5552.056817375968, pvalue=0.0)
```

Conclusion:

Since the p-value is 0, there is a statistically significant difference in the mean Commitment Term for at least two of the non-"M" classes. Because of this, we will proceed to fit the model with the dummy variable of Charged Class (performed earlier in the Introduction and Data Cleaning notebook).

Court Name

The court name does not appear to have a large impact on the median commitment term; however, the top sentences seem to be in Chicago and Maywood. An ANOVA test will be conducted to determine if at least one court has a statistical difference in sentencing term.

Null Hypothesis: There are no differences between all means.

In [363…
```
category = data['SENTENCE_COURT_NAME'].unique()
print_means(data,'SENTENCE_COURT_NAME',category)
st.f_oneway(data[data['SENTENCE_COURT_NAME']==category[0]]['COMMITMENT_TERM'
           data[data['SENTENCE_COURT_NAME']==category[1]]['COMMITMENT_TERM']
           data[data['SENTENCE_COURT_NAME']==category[2]]['COMMITMENT_TERM']
           data[data['SENTENCE_COURT_NAME']==category[3]]['COMMITMENT_TERM']
           data[data['SENTENCE_COURT_NAME']==category[4]]['COMMITMENT_TERM']
           data[data['SENTENCE_COURT_NAME']==category[5]]['COMMITMENT_TERM']
```

```
District 2 - Skokie: 1367.8749241044322
District 1 - Chicago: 1736.0061093034067
District 5 - Bridgeview: 1303.4711695645713
District 6 - Markham: 1616.3856930587986
District 3 - Rolling Meadows: 1452.3862617203424
District 4 - Maywood: 1521.2032949953373
```

Out[363]:
```
F_onewayResult(statistic=96.25893462156853, pvalue=2.4245748896734982e-101)
```

Conclusion:

Since the p-value is extremely small, there does appear to be at least two courts that have a statistically significant difference in sentencing means. Because of this, we will proceed to fit the model with the dummy variable of Sentencing Courts (performed earlier in the Introduction and Data Cleaning notebook).

Race

Using the boxplot, American Indian race appears to have the highest median sentencing terms. Biracial and Hispanic appear to have the lowest sentencing terms. All other races appear to have similar sentencing terms. An ANOVA test will be conducted to determine if at least one race has a statistical difference in sentencing term.

Null Hypothesis: There are no differences between all means.

In [350…

```
category = data['RACE'].unique()
print_means(data,'RACE',category)
st.f_oneway(data[data['RACE']==category[0]]['COMMITMENT_TERM'].values,
           data[data['RACE']==category[1]]['COMMITMENT_TERM'].values,
           data[data['RACE']==category[2]]['COMMITMENT_TERM'].values,
           data[data['RACE']==category[3]]['COMMITMENT_TERM'].values,
           data[data['RACE']==category[4]]['COMMITMENT_TERM'].values,
           data[data['RACE']==category[5]]['COMMITMENT_TERM'].values,
           data[data['RACE']==category[6]]['COMMITMENT_TERM'].values,
           data[data['RACE']==category[7]]['COMMITMENT_TERM'].values)
```

```
White: 1469.3877695576894
Black: 1612.1833451157624
White [Hispanic or Latino]: 1630.0113991819057
American Indian: 1823.9285714285713
Asian: 1708.1559405940593
HISPANIC: 1056.3563829787233
White/Black [Hispanic or Latino]: 1679.740409207161
Biracial: 940.8333333333334
```

Out[350]:

```
F_onewayResult(statistic=12.099252521039638, pvalue=1.5576607013177494e-15)
```

Conclusion:

There appear to be at least two races that have different mean sentencing times. Because of this, we will proceed to fit the model with the dummy variable of Race (performed earlier in the Introduction and Data Cleaning notebook).

Gender

Using the boxplot, males seem to be sentenced longer than females. A t-test will be conducted to identify if the difference is significant.

Null Hypothesis: There are no differences between mean term length for males and females.

In [358…

```
print('Group 1:  Male, Group 2: Female')
ttest(data[data['GENDER']=='Male']['COMMITMENT_TERM'],
     data[data['GENDER']=='Female']['COMMITMENT_TERM'])
```

```
Group 1:  Male, Group 2: Female
Mean 1: 1617.9224935929312
Mean 2: 1195.1688700031477
TtestResult(statistic=12.402979746071498, pvalue=2.8045863028808094e-35, df=
55461.0)
```

Conclusion:

Males appear to have a higher mean sentencing time. Because of this, we will proceed to fit the model with the dummy variable of Gender (performed earlier in the Introduction and Data Cleaning notebook).

Judge Group

Using the boxplot, there does not appear to be any differences in the median sentencing terms across Judge Groups. An ANOVA test will be conducted to determine if at least one judge group has a statistical difference in sentencing term.

Null Hypothesis: There are no differences between all means.

In [365…

```
category = data['Judge_Group'].unique()
print_means(data,'Judge_Group',category)
st.f_oneway(data[data['Judge_Group']==category[0]]['COMMITMENT_TERM'].values
           data[data['Judge_Group']==category[1]]['COMMITMENT_TERM'].values,
           data[data['Judge_Group']==category[2]]['COMMITMENT_TERM'].values,
           data[data['Judge_Group']==category[3]]['COMMITMENT_TERM'].values)
```

```
Oldest: 1595.8333449671925
New: 1568.7591606133979
Moderate: 1632.101761034329
Newest: 1533.2317073170732
```

Out[365]:
```
F_onewayResult(statistic=1.6490677686763868, pvalue=0.17571741861345905)
```

Conclusion:

With a p-value > 0.05, there does not appear to be a Judge group with a statistically significant different mean from another group. Because of this, removing this categorical feature from the model may be considered upon confirmation from the feature importance analysis and multicollinearity check.

Incident In Chicago

Incidents in Chicago seem to have a higher top end of sentencing; however, the median appears to be similar with incidents out of Chicago. A t-test will be conducted to identify if the difference is significant.

Null Hypothesis: There are no differences between mean term length for incidents in Chicago and out of Chicago.

In [366…
```
print('Group 1:  In Chicago, Group 2: Out of Chicago')
ttest(data[data['Incident_Chicago']==1]['COMMITMENT_TERM'],
     data[data['Incident_Chicago']==0]['COMMITMENT_TERM'])
```

```
Group 1:  In Chicago, Group 2: Out of Chicago
Mean 1: 1640.6210814459278
Mean 2: 1472.325367052584
TtestResult(statistic=9.521893117678093, pvalue=1.7660829985875906e-21, df=5
5461.0)
```

Conclusion:

Incidents in Chicago appear to have a higher mean sentencing time. Because of this, we will proceed to fit the model with the dummy variable of Incident Chicago (performed earlier in the Introduction and Data Cleaning notebook).

EDA: Correlation Matrix

Due to the vast numbers of dummy variables, only the top 20 correlated features will be included in the pairplot.

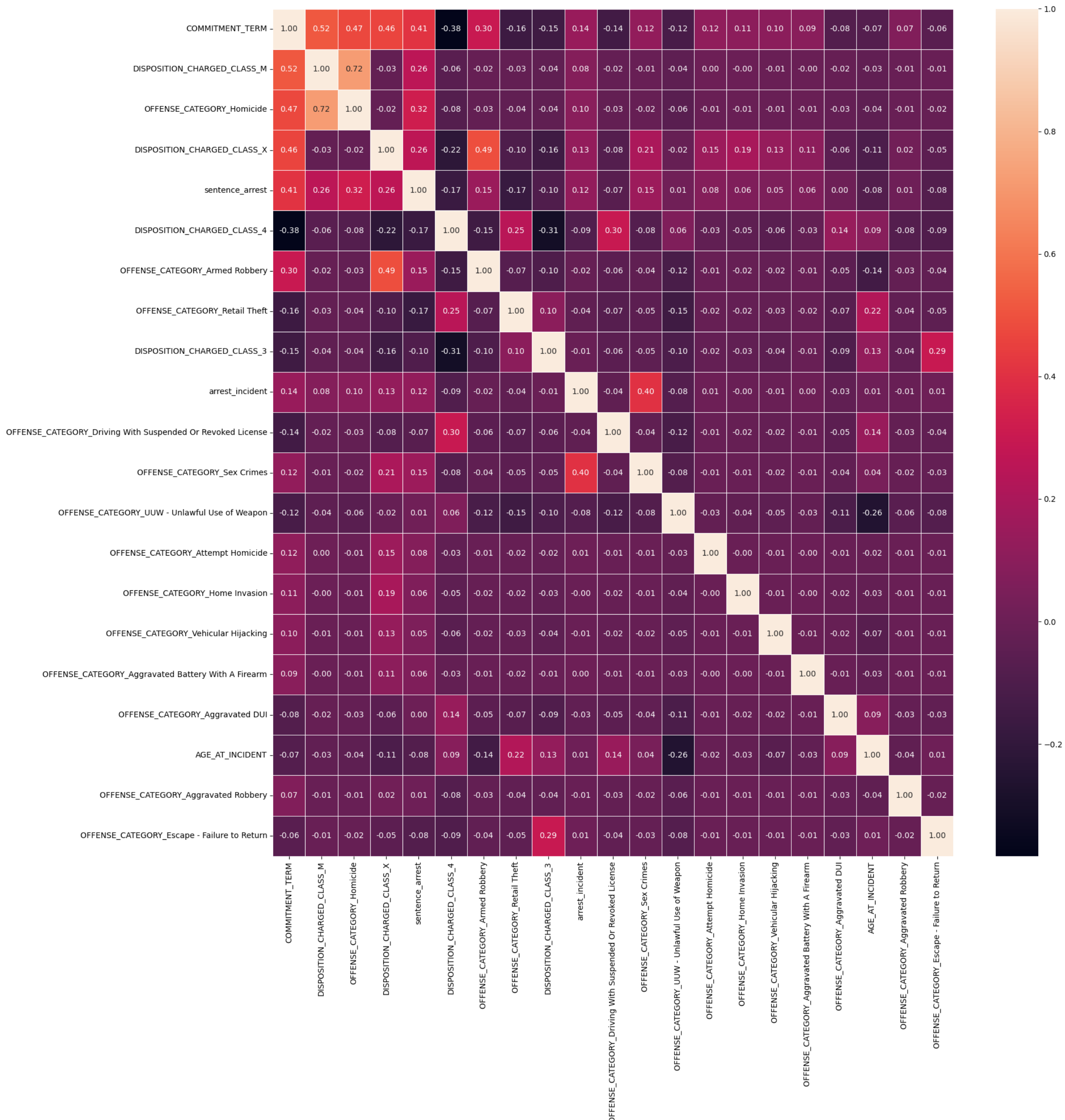
- All categorical cariables were first converted to dummy variables. See the Introduction and Data Cleaning notebook for more details.

In [370…

```
corr_df = data_dummy.corr()['COMMITMENT_TERM'].sort_values(ascending = False
```

In [385…

```
plt.figure(figsize = (20,20))
sns.heatmap(data_dummy[['COMMITMENT_TERM']+(abs(corr_df).sort_values(ascendi
            annot=True, fmt=".2f",
            linewidth=.5)
plt.tight_layout()
```



As we saw in the analysis of continuous variables, the time between sentence and arrest shows a moderate positive correlation with term length. Surprisingly, Charged Class 4 has a moderate negative correlation with term length.

Charge Class "M" and Offense Category Homicide appear to both be moderately correlated with term length. This makes sense as homicides and first-degree murder are often related. This can be seen through the 0.72 correlation between the two. This indicates there is collinearity between the two variables.

No other variables appear highly correlated with each other; however, multicollinearity testing will be conducted to confirm.

Summary: EDA Conclusion and Discussion

Out of the five continuous variables, the duration between arrest and sentence in days demonstrated a moderately strong positive linear relationship with prison sentence length. This suggests that longer periods between arrest and sentencing are associated with longer prison terms. Notably, there is no significant risk of multicollinearity among the continuous variables, as indicated by the low R^2 values across the pairplot and correlation matrix. The highest collinearity risk observed was between the YEAR variable and the arrest-sentence duration, with an R^2 of 0.36.

When examining the influence of the month of the year on arrest and disposition, certain months showed lower mean sentencing times. Arrests occurring between February and May, and dispositions in January and February, resulted in t-tests with p-values less than 0.05, indicating statistically significant differences. However, fitting a linear model using the sine and cosine transformations of the months did not reveal an overall linear relationship across all months. This suggests that while there may not be a consistent linear relationship between month and prison sentence lengths, certain months may exhibit minor biases affecting term lengths.

Within the categorical variables, the ANOVA tests conducted to identify differences in mean sentencing times across Offense Category, Charged Class, Court Names, and Race resulted in a p-value less than 0.05. The ANOVA test conducted on the Judge Group, unsurprisingly, resulted in a p-value of 0.175. This could be seen similarly with the box plots of the distribution of sentence terms without outliers. The Judge Groups with the highest delta between mean sentence times were the Newest Judges and the Moderate Judges. This is a reasonable conclusion as this feature was engineered with incomplete data. Since the judge groups were created on a subsection of the population of cases, it may not reflect the reality of clerkship in Cook County, which includes term length, township served, midterm vacancies, etc.

Both t-tests identified significant differences in mean sentence times for incidents in Chicago and for the gender of the participant. The delta in mean sentence times was quite large between Males and Females. This could be an indication of feature importance, but will be confirmed through further analysis.

The sentence term length had a few features with an absolute correlation greater than

0.3. These include Charged Class of M, Offense Category of Homicide, Charged Class X, Sentence - Arrest days, Charged Class of 4, and Offense Category of Armed Robbery. This seems quite reasonable as Charged Class M and Charged Class X are the top two severe felonies. There does appear to be collinearity between Charged Class M and Offense Category Homicide. This is in alignment with Cook County guidelines as First Degree Murder and Homicides should be highly correlated.

Due to the large number of unique values in some categorical features, each variable within the feature may have weak statistical power due to the lack of sample. This may lead to incomplete conclusions regarding ANOVA or t-tests. Future iterations of this project could heavily benefit from exogenous variables enriching the current list of features or perhaps engineering new features. Some of these columns with potential for improvement in detail would be the sentencing court, judge name, participant information, and victim information.

Model Selection: Multicollinearity

- VIF (Variance Inflation Factor)

For the Multi Linear Regression model, we must account for multicollinearity in the model and remove any features with a high VIF and/or correlation with another.

In [32]:
```
X = data_dummy.drop('COMMITMENT_TERM', axis = 1).copy()
```

In [401…
```
vif_df = add_constant(X).copy() # add constant to account for response varia

vif_df_results = pd.Series([variance_inflation_factor(vif_df.values, i)
               for i in range(vif_df.shape[1])],
              index=vif_df.columns)
```

In [408…
```
vif_df_results.sort_values(ascending = False)[1:21] # ignore constant
```

```
Out[408]: RACE_Black                                                    405.708956
RACE_White [Hispanic or Latino]                               253.791998
RACE_White                                                    202.209268
OFFENSE_CATEGORY_UUW - Unlawful Use of Weapon                  57.940244
OFFENSE_CATEGORY_Burglary                                      32.044423
OFFENSE_CATEGORY_Retail Theft                                  28.661473
RACE_HISPANIC                                                  21.013366
OFFENSE_CATEGORY_Driving With Suspended Or Revoked License     19.608309
OFFENSE_CATEGORY_Armed Robbery                                 19.099974
OFFENSE_CATEGORY_Aggravated DUI                                16.225142
OFFENSE_CATEGORY_Residential Burglary                          16.206953
RACE_White/Black [Hispanic or Latino]                          14.890344
OFFENSE_CATEGORY_Narcotics                                     12.557861
OFFENSE_CATEGORY_Robbery                                       12.257635
OFFENSE_CATEGORY_Aggravated Battery Police Officer             11.254455
OFFENSE_CATEGORY_Possession of Stolen Motor Vehicle            11.165375
OFFENSE_CATEGORY_Sex Crimes                                    10.727202
OFFENSE_CATEGORY_Theft                                         10.605497
OFFENSE_CATEGORY_Escape - Failure to Return                     8.823136
OFFENSE_CATEGORY_Failure to Register as a Sex Offender          8.821002
dtype: float64
```

```
In [412…: data_dummy.columns[data_dummy.columns.str.contains('RACE')]
```

```
Out[412]: Index(['RACE_Asian', 'RACE_Biracial', 'RACE_Black', 'RACE_HISPANIC',
       'RACE_White', 'RACE_White [Hispanic or Latino]',
       'RACE_White/Black [Hispanic or Latino]'],
      dtype='object')
```

```
In [413…: data_dummy.columns[data_dummy.columns.str.contains('OFFENSE_CATEGORY')]
```

```
Out[413]: Index(['OFFENSE_CATEGORY_Aggravated Assault Police Officer Firearm',
       'OFFENSE_CATEGORY_Aggravated Battery',
       'OFFENSE_CATEGORY_Aggravated Battery Police Officer',
       'OFFENSE_CATEGORY_Aggravated Battery Police Officer Firearm',
       'OFFENSE_CATEGORY_Aggravated Battery With A Firearm',
       'OFFENSE_CATEGORY_Aggravated DUI',
       'OFFENSE_CATEGORY_Aggravated Discharge Firearm',
       'OFFENSE_CATEGORY_Aggravated Fleeing and Eluding',
       'OFFENSE_CATEGORY_Aggravated Identity Theft',
       'OFFENSE_CATEGORY_Aggravated Robbery',
       'OFFENSE_CATEGORY_Aggravated Robbery BB Gun',
       'OFFENSE_CATEGORY_Armed Robbery', 'OFFENSE_CATEGORY_Armed Violence',
       'OFFENSE_CATEGORY_Arson', 'OFFENSE_CATEGORY_Arson and Attempt Arso
n',
       'OFFENSE_CATEGORY_Attempt Armed Robbery',
       'OFFENSE_CATEGORY_Attempt Arson', 'OFFENSE_CATEGORY_Attempt Homicid
e',
       'OFFENSE_CATEGORY_Attempt Sex Crimes',
       'OFFENSE_CATEGORY_Attempt Vehicular Hijacking',
       'OFFENSE_CATEGORY_Battery', 'OFFENSE_CATEGORY_Bomb Threat',
       'OFFENSE_CATEGORY_Bribery', 'OFFENSE_CATEGORY_Burglary',
       'OFFENSE_CATEGORY_Child Abduction',
       'OFFENSE_CATEGORY_Child Pornography',
       'OFFENSE_CATEGORY_Communicating With Witness',
       'OFFENSE_CATEGORY_Credit Card Cases',
       'OFFENSE_CATEGORY_Criminal Damage to Property',
```

```
       'OFFENSE_CATEGORY_Criminal Trespass To Residence',
       'OFFENSE_CATEGORY_DUI', 'OFFENSE_CATEGORY_Deceptive Practice',
       'OFFENSE_CATEGORY_Disarming Police Officer',
       'OFFENSE_CATEGORY_Dog Fighting', 'OFFENSE_CATEGORY_Domestic Batter
y',
       'OFFENSE_CATEGORY_Driving With Suspended Or Revoked License',
       'OFFENSE_CATEGORY_Escape - Failure to Return',
       'OFFENSE_CATEGORY_Failure To Pay Child Support',
       'OFFENSE_CATEGORY_Failure to Register as a Sex Offender',
       'OFFENSE_CATEGORY_Forgery', 'OFFENSE_CATEGORY_Fraud',
       'OFFENSE_CATEGORY_Fraudulent ID', 'OFFENSE_CATEGORY_Gambling',
       'OFFENSE_CATEGORY_Gun - Non UUW', 'OFFENSE_CATEGORY_Gun Running',
       'OFFENSE_CATEGORY_Hate Crimes', 'OFFENSE_CATEGORY_Home Invasion',
       'OFFENSE_CATEGORY_Homicide', 'OFFENSE_CATEGORY_Human Trafficking',
       'OFFENSE_CATEGORY_Identity Theft',
       'OFFENSE_CATEGORY_Impersonating Police Officer',
       'OFFENSE_CATEGORY_Intimidation', 'OFFENSE_CATEGORY_Kidnapping',
       'OFFENSE_CATEGORY_Major Accidents', 'OFFENSE_CATEGORY_Narcotics',
       'OFFENSE_CATEGORY_Obstructing Justice',
       'OFFENSE_CATEGORY_Other Offense', 'OFFENSE_CATEGORY_PROMIS Conversio
n',
       'OFFENSE_CATEGORY_Pandering', 'OFFENSE_CATEGORY_Perjury',
       'OFFENSE_CATEGORY_Police Shooting',
       'OFFENSE_CATEGORY_Possession Of Burglary Tools',
       'OFFENSE_CATEGORY_Possession of Contraband in Penal Institution',
       'OFFENSE_CATEGORY_Possession of Shank in Penal Institution',
       'OFFENSE_CATEGORY_Possession of Stolen Motor Vehicle',
       'OFFENSE_CATEGORY_Prostitution',
       'OFFENSE_CATEGORY_Reckless Discharge of Firearm',
       'OFFENSE_CATEGORY_Reckless Homicide',
       'OFFENSE_CATEGORY_Residential Burglary',
       'OFFENSE_CATEGORY_Retail Theft', 'OFFENSE_CATEGORY_Robbery',
       'OFFENSE_CATEGORY_Sex Crimes', 'OFFENSE_CATEGORY_Stalking',
       'OFFENSE_CATEGORY_Tampering', 'OFFENSE_CATEGORY_Theft',
       'OFFENSE_CATEGORY_Theft by Deception',
       'OFFENSE_CATEGORY_UUW - Unlawful Use of Weapon',
       'OFFENSE_CATEGORY_Unlawful Restraint',
       'OFFENSE_CATEGORY_Vehicular Hijacking',
       'OFFENSE_CATEGORY_Vehicular Invasion',
       'OFFENSE_CATEGORY_Violate Bail Bond',
       'OFFENSE_CATEGORY_Violation Order Of Protection',
       'OFFENSE_CATEGORY_Violation of Sex Offender Registration'],
      dtype='object')
```

We can observe that within the top 20 VIF scores, there are high VIF scores in the RACE categorical variable. Five out of the seven RACE columns are included. OFFENSE CATEGORY is another categorical variable that is showing high multicollinearity. Fifteen out of the top 20 are from this category. Although a good amount of the columns related to OFFENSE CATEGORY do not have high VIF scores. Any column with a VIF score > 10 will be dropped, and the VIF will be recalculated.

In [422…
```
vif_df_2 = add_constant(vif_df.drop(vif_df_results[vif_df_results>10].index,
vif_df_results2 = pd.Series([variance_inflation_factor(vif_df_2.values, i)
               for i in range(vif_df_2.shape[1])],
              index=vif_df_2.columns)
```

In [424…
```
vif_df_results2[vif_df_results2>10].sort_values(ascending = False)
```

Out[424]:
```
const     515377.689317
dtype: float64
```

It appears that all columns have a VIF score < 10. These non-multicollinear columns will be fit into the linear regression model. Since tree-based models can handle multicollinearity much better than the linear regression model, due to the splitting of the feature space based on individual features, the full dummy variable features will first be passed prior to checking for feature importance.

Model Selection: Stationarity

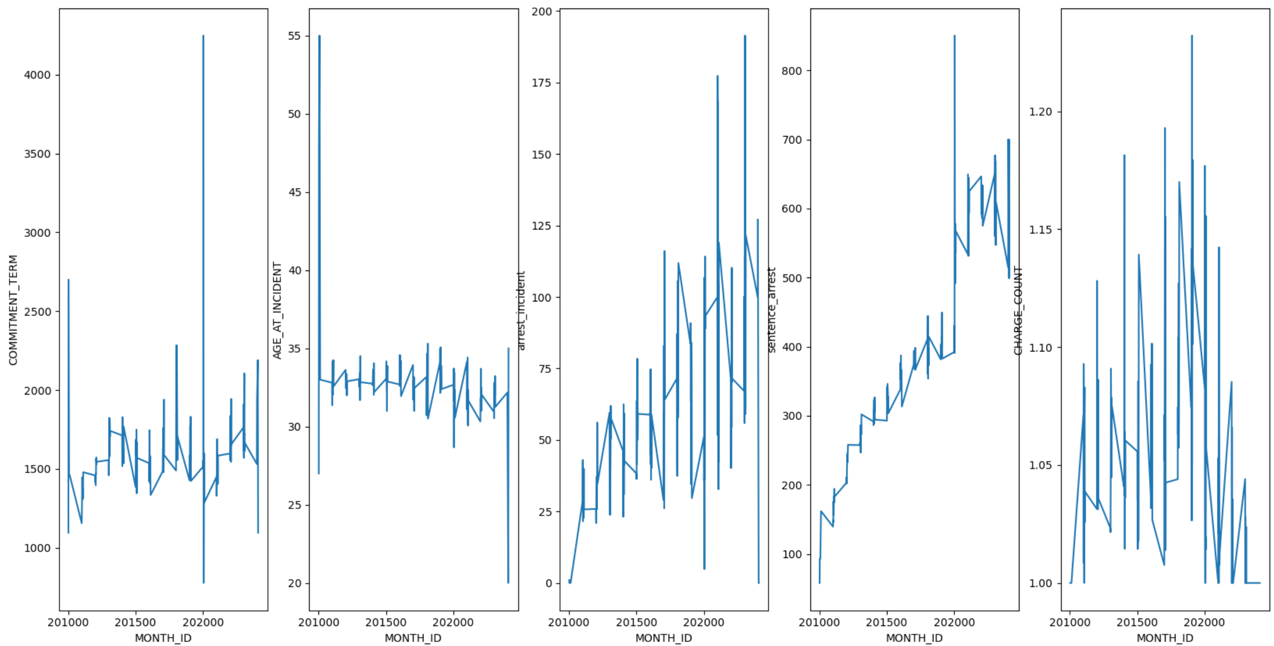
- Using the ADFuller test for stationarity on non-categorical columns.

Due to the large dataset, each of the continuous columns will be aggregated by sentence year + disposition month, and the mean values will be tested for stationarity.

In [493…
```
def stationary(df1,testing):
    print(testing+' results: '+ str(adfuller(df1)))

def s_test(df):
    fig,ax = plt.subplots(nrows = 1, ncols =5, figsize=(20, 10))
    col = df.columns
    for x in range(len(col)):
        stationary(df[col[x]],col[x])
        sns.lineplot(df[col[x]], ax=ax[x])

station_df = data[['COMMITMENT_TERM']+cont_var+['sin_DISPOSITION','cos_DISPO
# Decoding Sine and Cosine month of year
for x in range(1,13):
    sin = np.sin(2 * np.pi * x/12)
    cos = np.cos(2 * np.pi * x/12)
    disp_ind = station_df[(station_df['cos_DISPOSITION']==cos)&(station_df['
    station_df.loc[disp_ind,'Month']=x
station_df['MONTH_ID'] = station_df['YEAR']*100+station_df['Month']
station_df = station_df.drop(['cos_DISPOSITION','sin_DISPOSITION',
                'Month','YEAR'], axis = 1).groupby(['MONTH_ID']).mean()
```

In [494…
```
s_test(station_df)
```

```
COMMITMENT_TERM results: (-8.990905709896547, 6.8962689466341425e-15, 1, 16
2, {'1%': -3.471374345647024, '5%': -2.8795521079291966, '10%': -2.576373330
2850174}, 2105.3447872667048)
AGE_AT_INCIDENT results: (-3.101342393442766, 0.02645400472687119, 5, 158,
{'1%': -3.4724305215713156, '5%': -2.8800127935315465, '10%': -2.57661923089
2485}, 537.1329787724271)
arrest_incident results: (-2.938587335566314, 0.04104131420524555, 4, 159,
{'1%': -3.472161410886292, '5%': -2.8798954259680936, '10%': -2.576556582809
2245}, 1412.7359123679275)
sentence_arrest results: (-1.1751426926697435, 0.6841962703051058, 5, 158,
{'1%': -3.4724305215713156, '5%': -2.8800127935315465, '10%': -2.57661923089
2485}, 1589.6621368242604)
CHARGE_COUNT results: (-3.677263930190903, 0.004444563929363444, 3, 160, {'
1%': -3.4718957209472654, '5%': -2.8797795410156253, '10%': -2.576494726562
5}, -508.7278245445066)
```



The days between sentencing and arrest dates seem to not be stationary. This is apparent through both the ADFuller test p-value of 0.68, but also the graph shows a clear upward trend in the average days between sentence and arrest. However, since the predicted feature of term length is stationary, and the arrest-sentence appears to have a positive trend over time, the inclusion of a non-differenced arrest-sentence may capture information to help predict sentencing length.

Moreover, by including the year of disposition, the model has additional contextual information about when the arrest occurred. This allows the model to learn patterns or trends in the relationship between sentence-arrest across different years. Since there is no multicollinearity issue between sentence-arrest and the year of disposition, I will include both features since the year of disposition accounts for some variability regarding sentence-arrest.

Machine Learning: Model Fitting

As mentioned in the Introduction and Data Cleaning notebook, the Linear Regression model is selected to provide a baseline performance. SVM would be a great choice since the features are mostly binary and categorical; however, the large dataset could mean that the performance would not be optimal.

Both the Random Forest regressor model and the XGBoost model were selected due to the models' ability to handle a large feature size (>120+) and a large dataset (>10K rows).

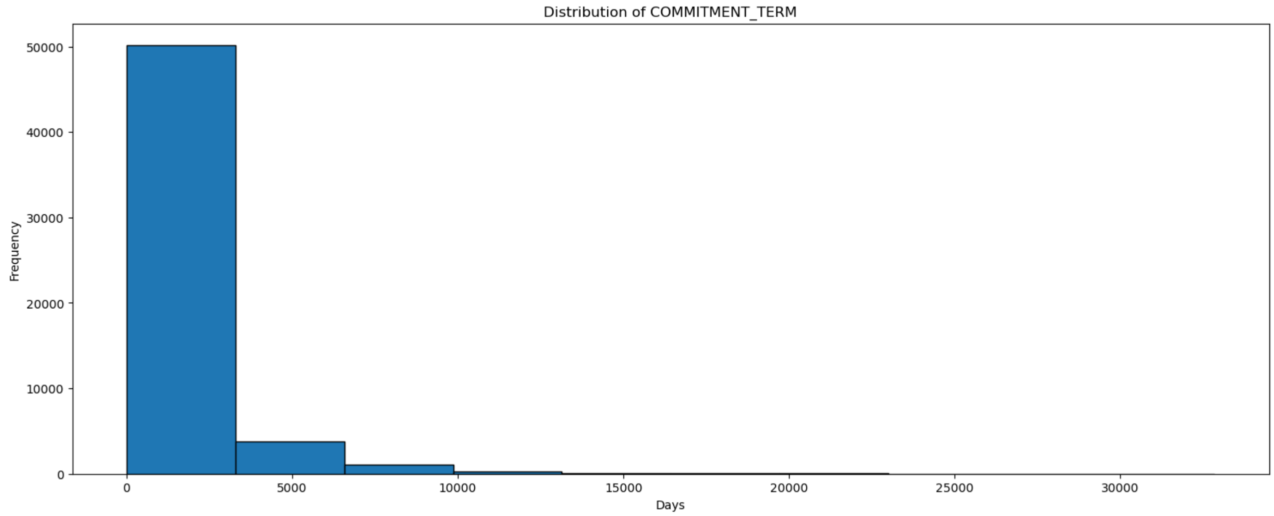
In [5]:

```
# Shape of data set
print(str(len(data_dummy))+' rows.\n'+str(len(data_dummy.columns))+' columns
print(str(len(data_dummy.columns)-1)+' features.')
plt.figure(figsize=(18, 15))
plt.subplot(2, 1, 1)
col = 'COMMITMENT_TERM'
data_dummy[col].plot(kind='hist', bins=10, edgecolor='black')
plt.xlabel('Days')
plt.title('Distribution of ' + col)
```

```
55463 rows.
121 columns.
120 features.
```

Out[5]:

```
Text(0.5, 1.0, 'Distribution of COMMITMENT_TERM')
```



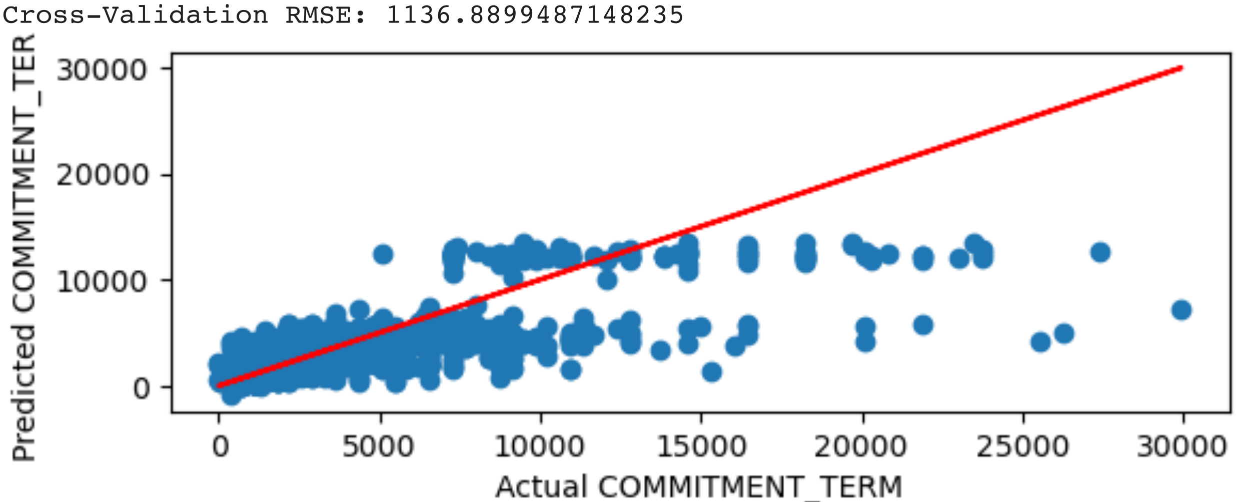
In [3]:
```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=-
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt
```

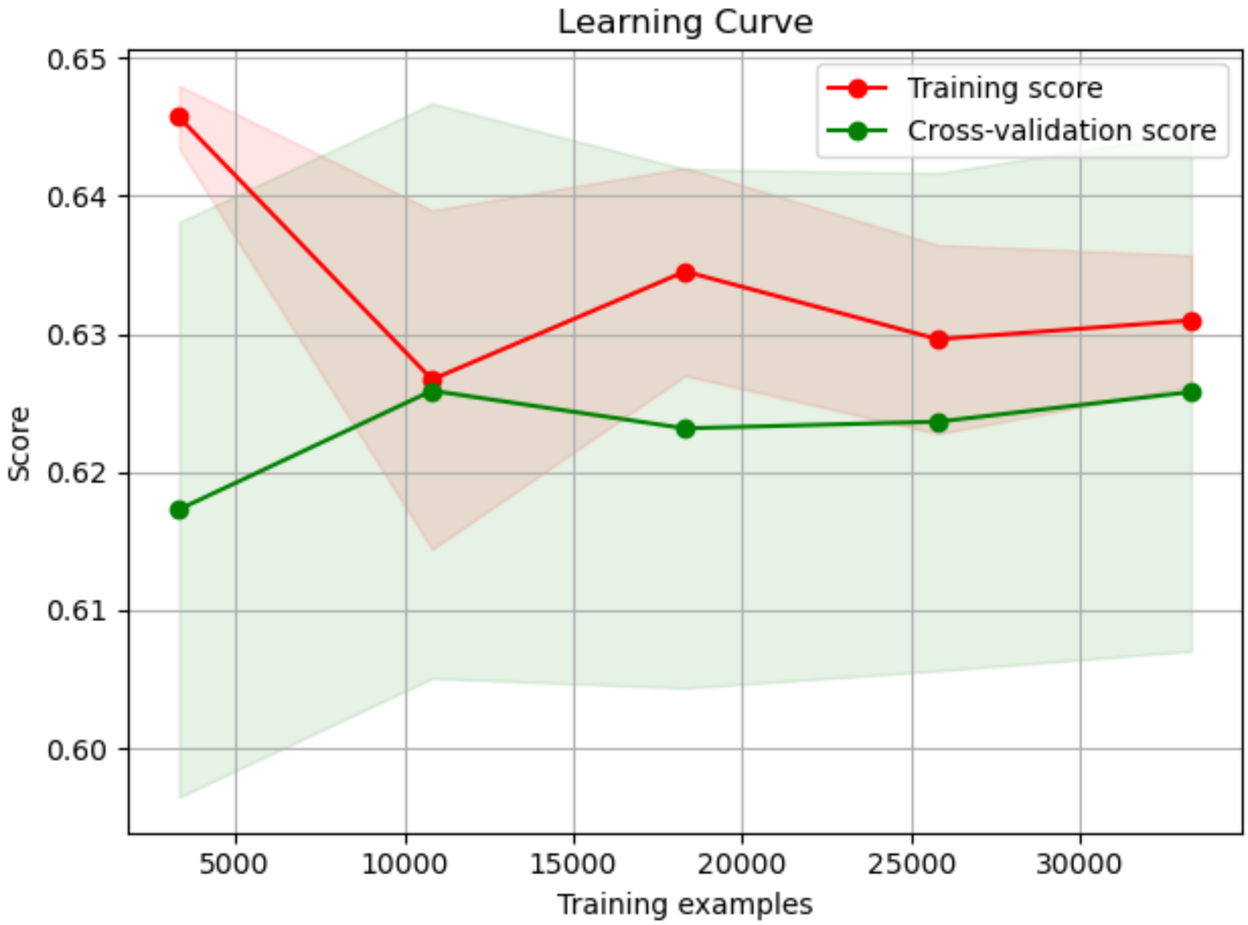
Linear Regression Model

In [511…

```
X = data_dummy.drop([col] + (vif_df_results[vif_df_results>10].drop('const')
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

lm = LinearRegression(n_jobs = -1).fit(X_train, y_train)
plm = lm.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(lm, X_train, y_train, cv=5, n_jobs = -1, scoring
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(lm, "Learning Curve", X_train, y_train, cv=5, n_jobs=-1)

plt.tight_layout()
plt.show()
```

```
MSE: 1259512.4070250215
RMSE: 1122.2800038426335
Relative RMSE: 0.697029585474447
MAE: 585.1811608901863
Relative MAE: 0.3634463597374551
EV: 0.64891309236881
R2: 0.648912217087557
Cross-Validation RMSE: 1136.8899487148235
```





Learning Curve
Training score
Cross-validation score
Score
Training examples
0.65
0.64
0.63
0.62
0.61
0.60
5000
10000
15000
20000
25000
30000

The linear regression model fits fairly well. The R2 score is 0.648, and the Root Mean Squared Error is 1122.28, with a relative RMSE of 69.7%. The Mean Absolute Error is 585, and the relative MAE is 36.3%. The cross-validated RMSE is similar to the RMSE, indicating that the model is not severely overfit.

Looking at the learning curve plots with the R2 score, we can see that the model starts with a higher training score and ends with a lower score, while the CV score starts lower and ends higher, converging with the training score. This shows that the model does not require more training examples.

The decrease in the training score suggests that the model fits the training data worse with more examples, indicating a reduction in overfitting. The increase in the CV score suggests that the model's ability to generalize to unseen data improves with more examples.

Nevertheless, the shaded area is much larger around the CV score. This indicates that the model suffers more from errors due to variance. This means the model is exhibiting low bias and high variance.

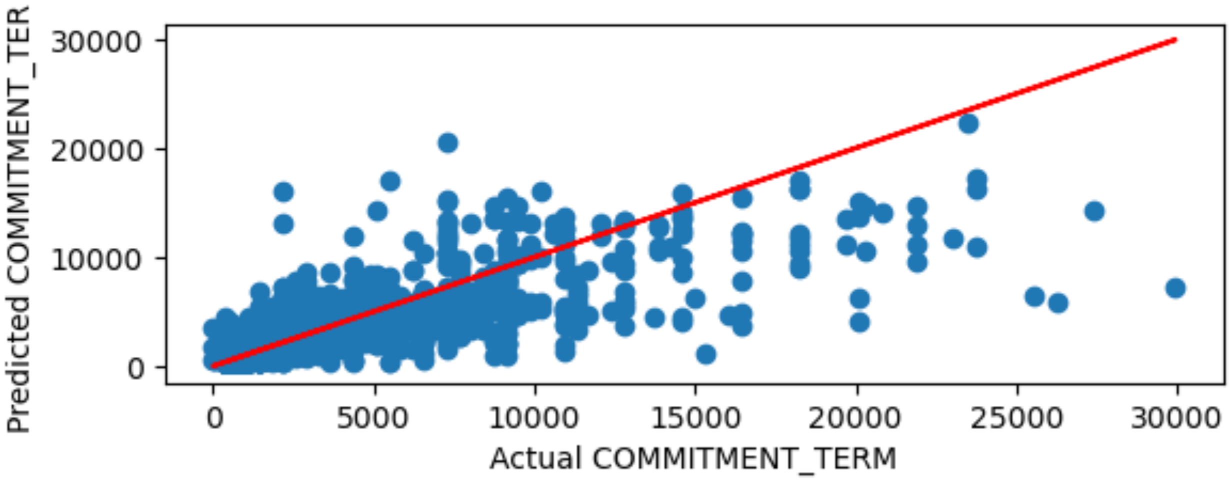
Random Forest Model

- Fit with default hyperparameters

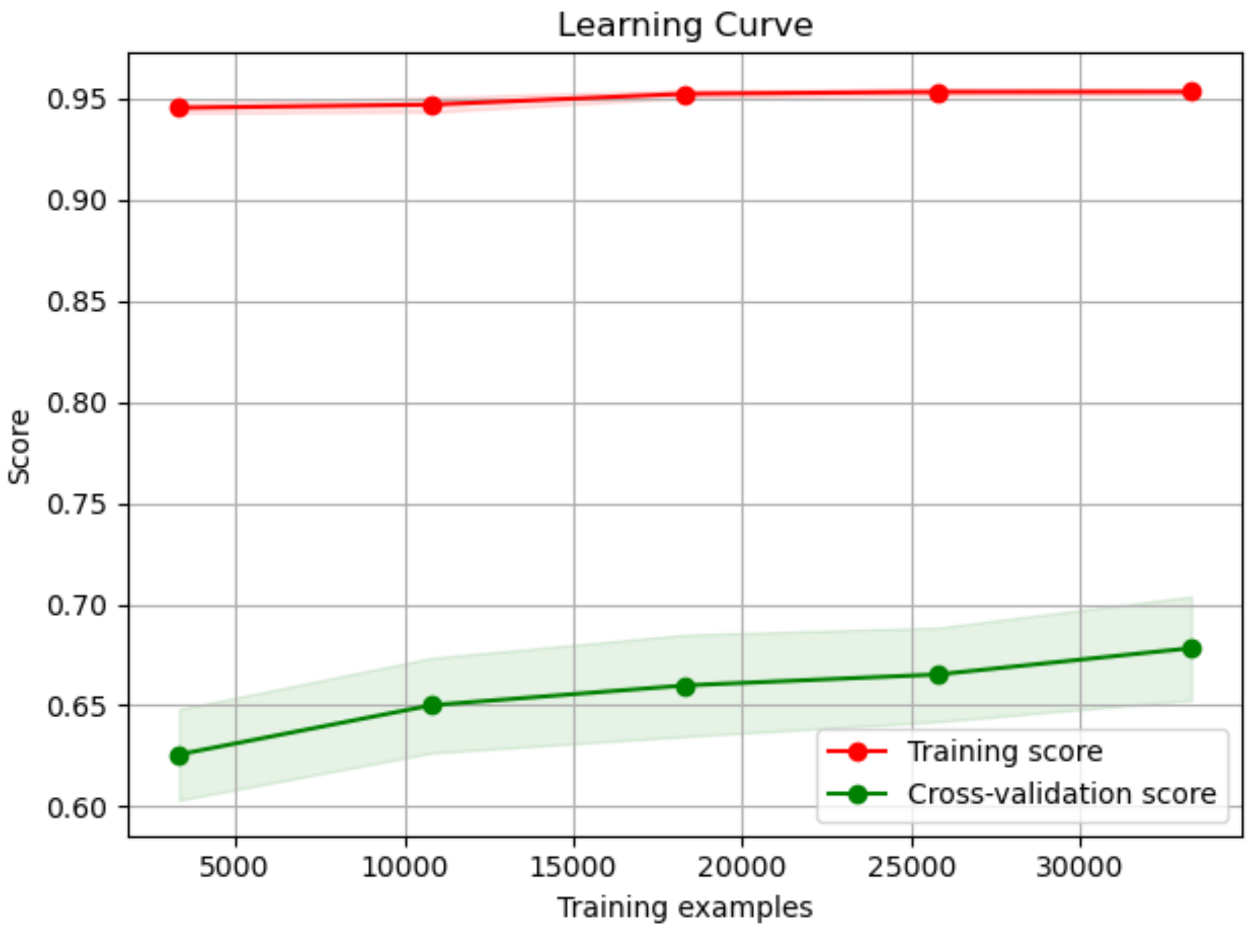
In [9]:

```
X = data_dummy.drop([col], axis = 1).values
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
rf = RandomForestRegressor(n_jobs = -1, random_state=0)
rf.fit(X_train, y_train)

# Predictions
plm = rf.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(rf, X_train, y_train, cv=5, n_jobs = -1, scoring
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(rf, "Learning Curve", X_train, y_train, cv=5, n_jobs=-1)

plt.tight_layout()
plt.show()
```

```
MSE: 1102306.678001323
RMSE: 1049.90793786947
Relative RMSE: 0.65208048990785
MAE: 494.9072311686142
Relative MAE: 0.30737871209379275
EV: 0.6929752049409452
R2: 0.6927331517256134
Cross-Validation RMSE: 1053.478392456391
```



Predicted COMMITMENT_TER
Actual COMMITMENT_TERM



Learning Curve
Score
Training examples
Training score
Cross-validation score

The random forest regressor model is fitted quite well. The R2 score is 0.69, and the Root Mean Squared Error is 1049.91, with a relative RMSE of 65.2%. The Mean Absolute Error is 494.91, and the relative MAE is 30.7%. The cross-validated RMSE is similar to the RMSE, indicating that the model is not severely overfit.

Looking at the learning curve plots with the R2 score, we can see that the model starts with a higher training score and ends flat, while the CV score starts lower and ends higher, but the two curves never converge. This suggests that the model may not be complex enough for the dataset and would benefit from more training data.

The gap between the training and CV scores indicates that the model, with the current settings, is overfitting the training data, and the CV score is unlikely to increase without more training or different hyperparameters. Thus, it does not mean that given a different set of hyperparameters the same effect would occur. It only seems that given the current setting, the rate of convergence is relatively small, thus getting to 95% would probably require more training data or different hyperparameters.

The feature importance will be analyzed prior to the next iteration of the Random Forest model.

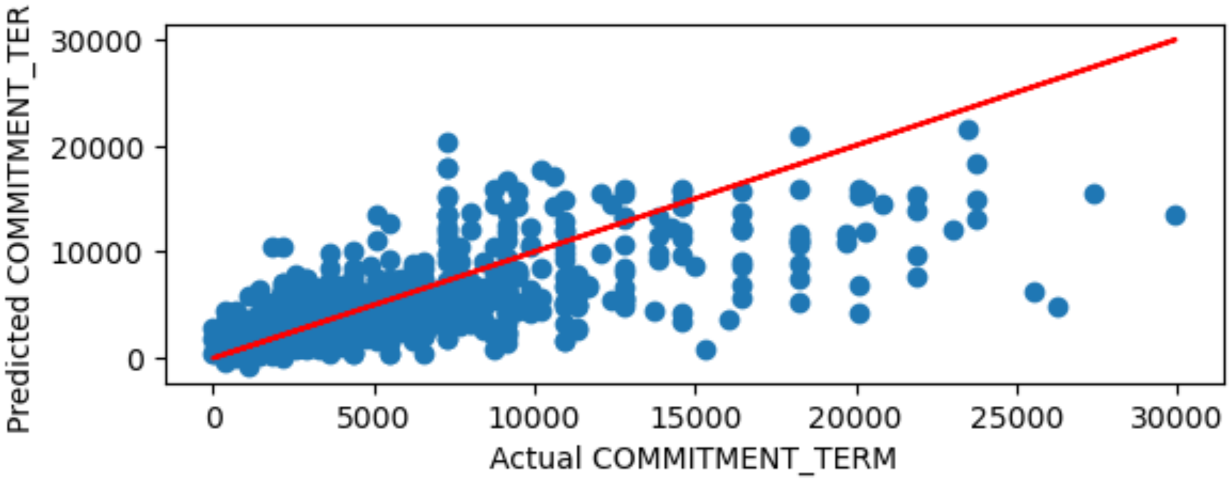
XGBoost Model

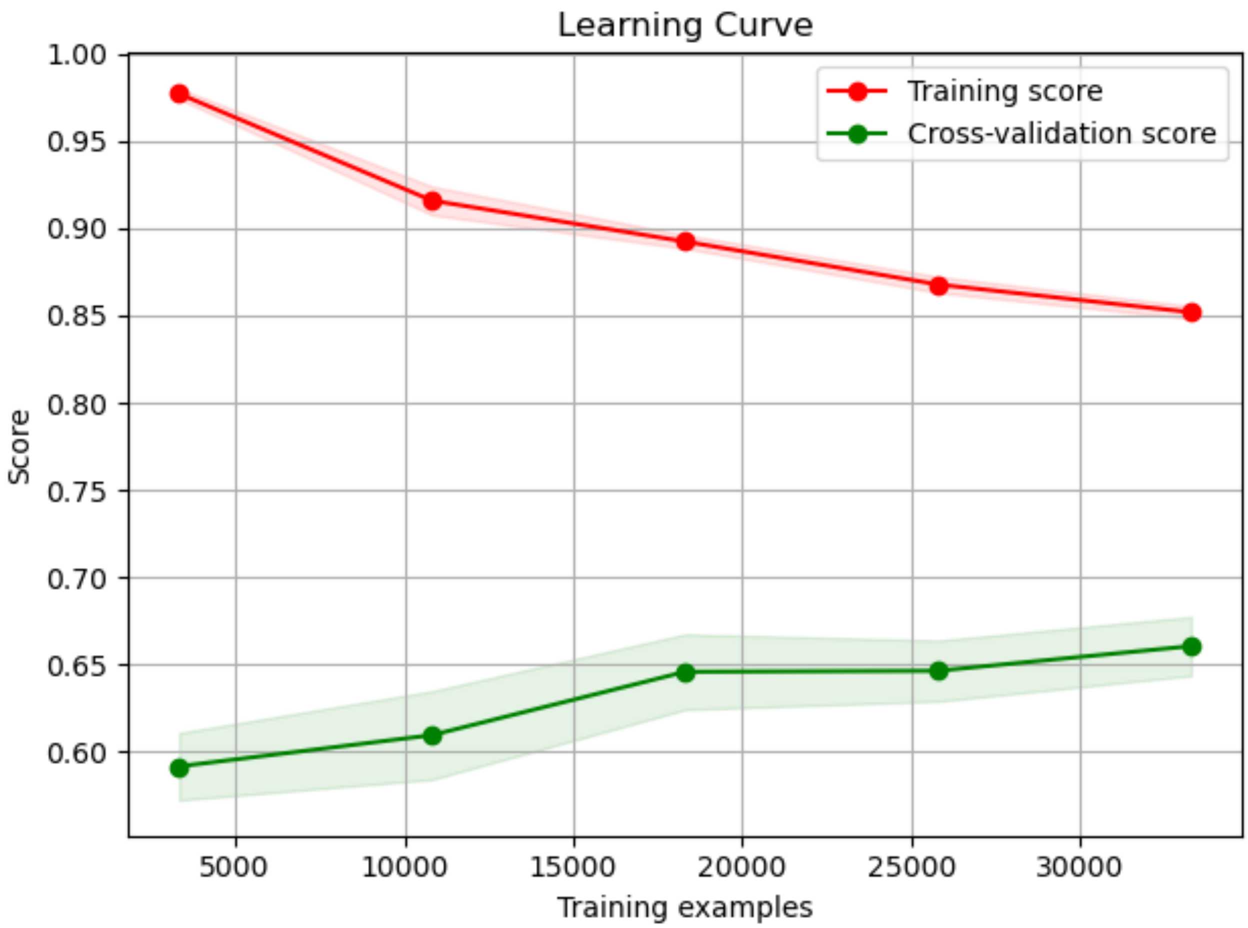
- Fit with default hyperparameters

In [6]:

```python
X = data_dummy.drop([col], axis = 1).values
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
xgb = XGBRegressor(n_jobs = -1, random_state=0)
xgb.fit(X_train, y_train)

# Predictions
plm = xgb.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(xgb, X_train, y_train, cv=5, n_jobs = -1, scorin
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(xgb, "Learning Curve", X_train, y_train, cv=5, n_jobs=-1

plt.tight_layout()
plt.show()
```

```
MSE: 1147668.610055112
RMSE: 1071.2929618246878
Relative RMSE: 0.6653623752945887
MAE: 515.7818337265383
Relative MAE: 0.32034358317594197
EV: 0.6800895579150485
R2: 0.6800885600053881
Cross-Validation RMSE: 1080.4867642606052
```


Predicted COMMITMENT_TER
Actual COMMITMENT_TERM
30000
20000
10000
0
5000
15000
25000


Learning Curve
Training score
Cross-validation score
Score
Training examples
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
5000
10000
15000
20000
25000
30000

The XGBoost model is also fitted quite well. The R2 score is 0.68, and the Root Mean Squared Error is 1071.29, with a relative RMSE of 66.54%. The Mean Absolute Error is 515.78, and the relative MAE is 32.03%. The cross-validated RMSE is similar to the RMSE, showing that the model is not severely overfit.

Looking at the learning curve plots with the R2 score, we can see that the model starts with a higher training score and ends lower, while the CV score starts lower and ends higher. The two curves never converge but are converging more than the RF model. This suggests that the model may not be complex enough for the dataset and would benefit from more training data.

The decrease in the training score suggests that the model fits the training data worse with more examples, an indication of reducing overfitting. The increase in the CV score suggests that the model's ability to generalize to unseen data improves with more examples. The shaded area around the CV score is larger than the training score's shaded area, indicating that the model suffers more from errors due to variance. This means the model is exhibiting low bias and high variance. However, the CV shaded area is reducing with training sample, showing signs the model is decreasing in variance.

Machine Learning: Ridge Regularization Linear Model

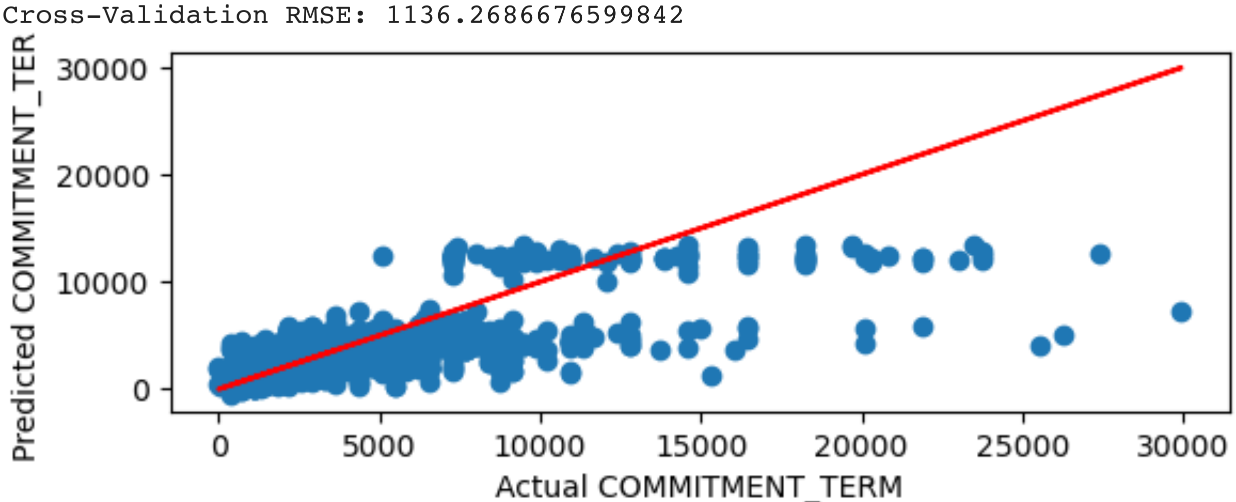
Ridge LM using regular data set

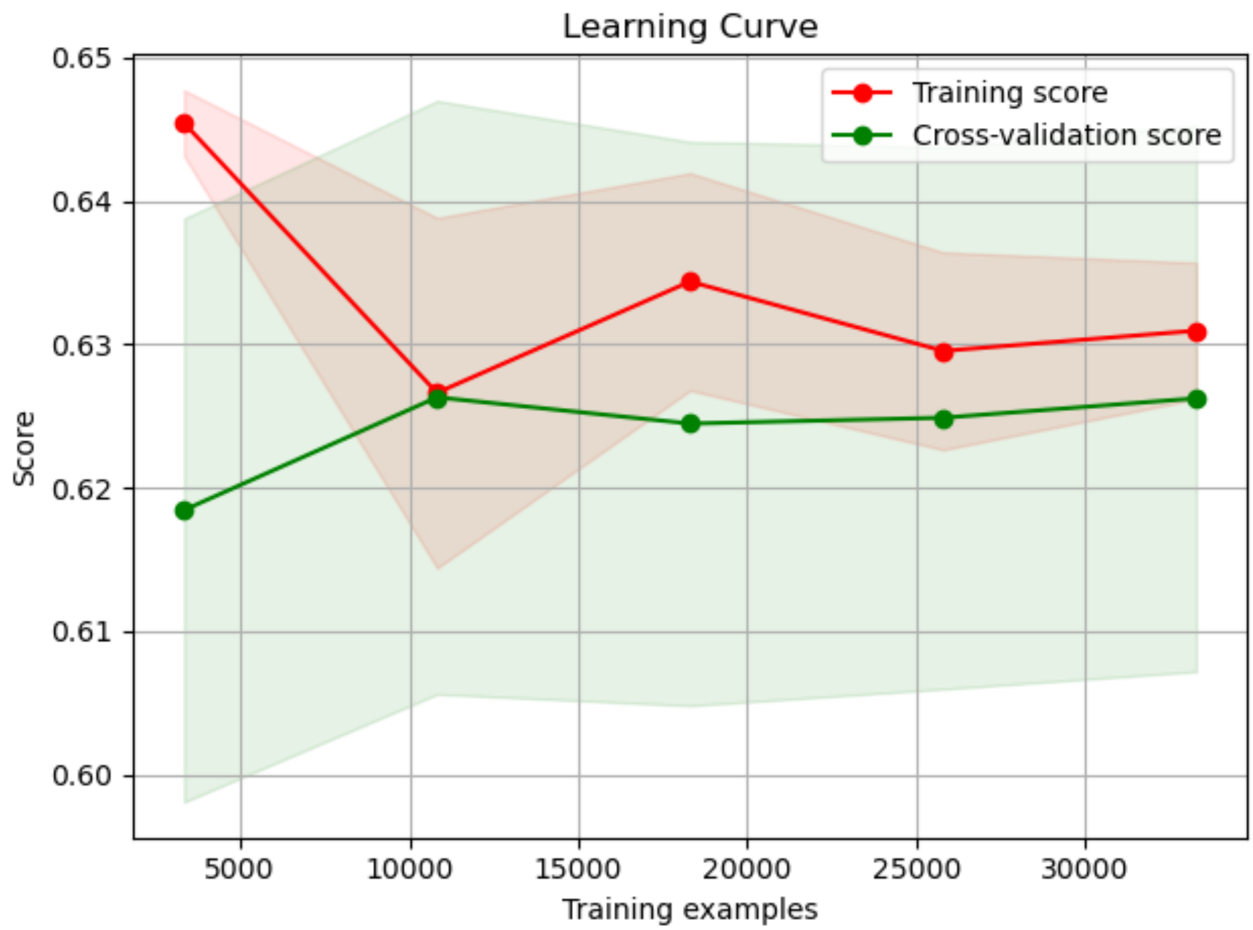
- Fit without multicollinear features

In [584…

```python
X = data_dummy.drop([col] + (vif_df_results[vif_df_results>10].drop('const')
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

rlm = Ridge(alpha = .75, random_state = 0,
            fit_intercept=True,solver='auto', max_iter=None).fit(X_train, y_
plm = rlm.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(rlm, X_train, y_train, cv=5, n_jobs = -1, scorin
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(rlm, "Learning Curve", X_train, y_train, cv=5, n_jobs=-1

plt.tight_layout()
plt.show()
```

```
MSE: 1258348.6440962818
RMSE: 1121.7614024810632
Relative RMSE: 0.6967074907290679
MAE: 585.0350636491056
Relative MAE: 0.36335562115257364
EV: 0.6492375283287548
R2: 0.6492366148022664
Cross-Validation RMSE: 1136.2686676599842
```



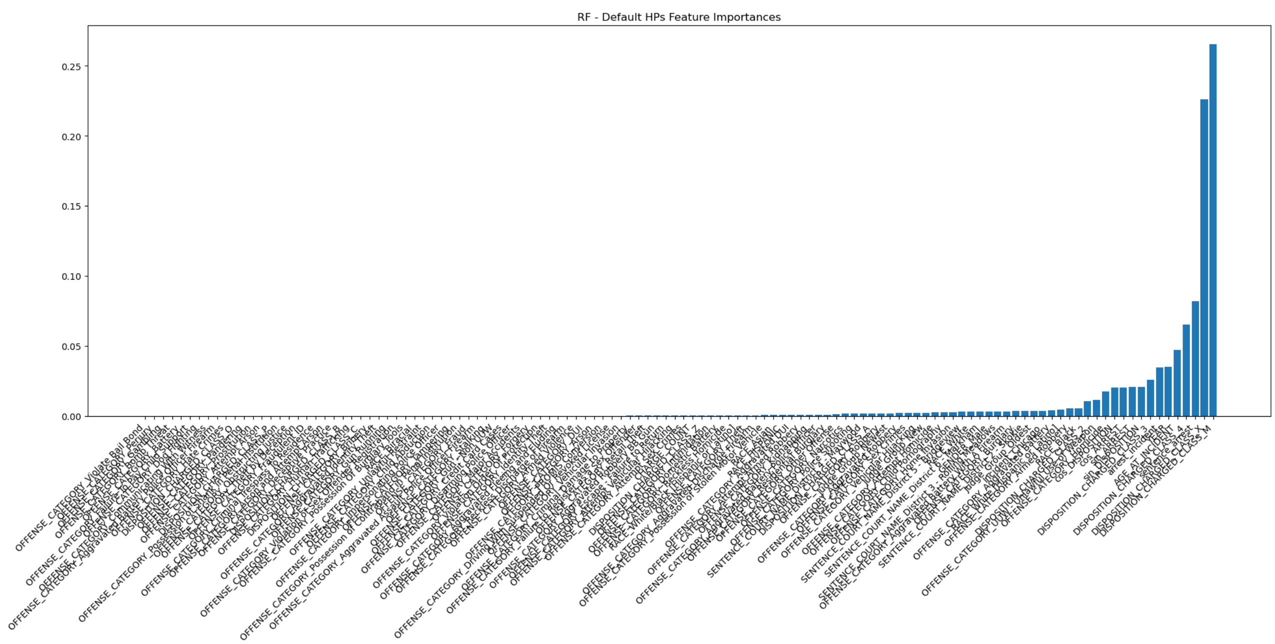


The results of the ridge regression show that the linear model was fit with minimal multicollinearity. The ridge model with an alpha of 0.75 has very similar metrics to the linear model, and the learning curve looks identical to the linear model's learning curve. This is common when the features do not have strong multicollinearity, so the ridge regression may not offer advantages over the linear regression model.

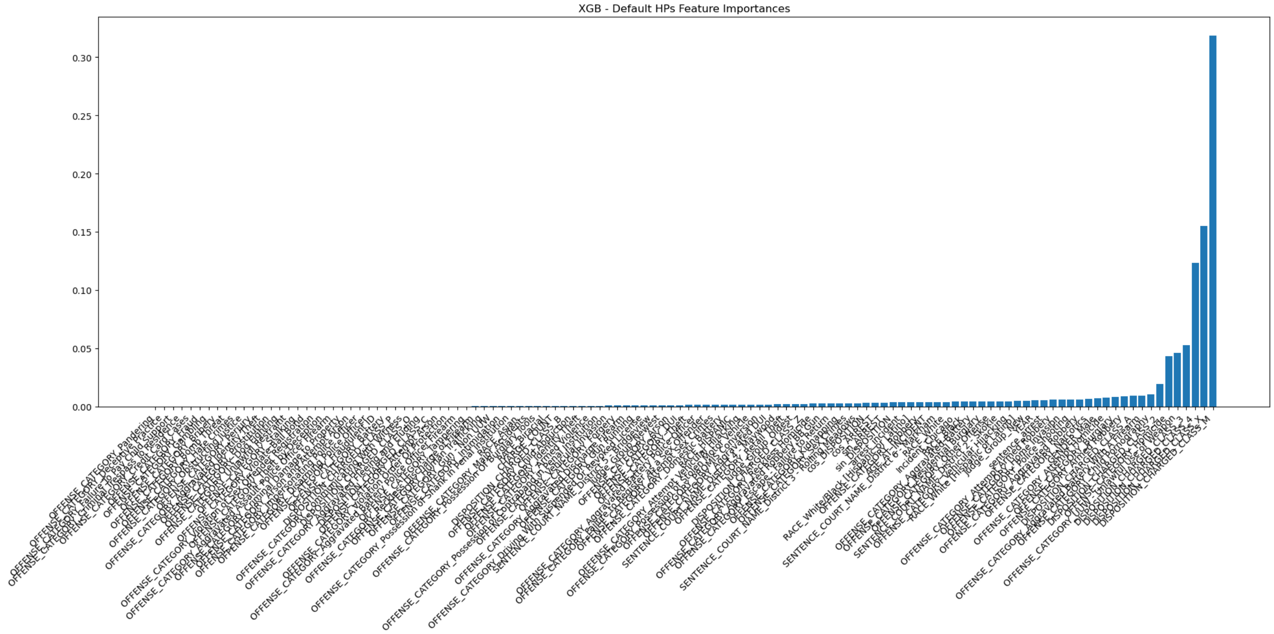
Machine Learning: Feature Importance

In [10]:
```python
def plot_feature_importance(model, feature_names, model_string):
    feature_importance = model.feature_importances_
    sorted_idx = np.argsort(feature_importance)
    x_ticks = np.arange(0, len(feature_names))
    fig, ax = plt.subplots(figsize=(20, 10))
    ax.bar(x_ticks, feature_importance[sorted_idx])
    ax.set_xticks(x_ticks)
    ax.set_xticklabels(np.array(feature_names)[sorted_idx], rotation=45, ha=
    ax.set_title(model_string + ' Feature Importances')
    plt.tight_layout()

    # Create DataFrame with feature names and importance scores
    feature_importance_df = pd.DataFrame({
        'Feature': np.array(feature_names)[sorted_idx],
        'Importance': feature_importance[sorted_idx]
    })

    return feature_importance_df
```

In [11]:
```python
rf_fi = plot_feature_importance(rf, data_dummy.drop([col], axis = 1).columns
                                'RF - Default HPs')
```



In [12]:
```python
xgb_fi = plot_feature_importance(xgb,data_dummy.drop([col], axis = 1).column
                                 'XGB - Default HPs')
```



It appears that the RF model has around 8 features that drive a huge amount of decision making. The XGB similarly has around 6 features.

PCA will be used to reduce the dimensionality of the non-top features.

Machine Learning: PCA to reduce feature dimension

- Apply PCA to reduce dimension of non top features

In [13]:
```
rf_fi.sort_values(by = 'Importance', ascending = False)[:8]
```

Out[13]:

	Feature	Importance
119	DISPOSITION_CHARGED_CLASS_M	0.265307
118	DISPOSITION_CHARGED_CLASS_X	0.225811
117	sentence_arrest	0.081830
116	DISPOSITION_CHARGED_CLASS_4	0.065210
115	AGE_AT_INCIDENT	0.047289
114	YEAR	0.035075
113	arrest_incident	0.034612
112	DISPOSITION_CHARGED_CLASS_3	0.025732

In [14]:
```
xgb_fi.sort_values(by = 'Importance', ascending = False)[:6]
```

Out[14]:

	Feature	Importance
119	DISPOSITION_CHARGED_CLASS_M	0.318390
118	DISPOSITION_CHARGED_CLASS_X	0.154829
117	DISPOSITION_CHARGED_CLASS_4	0.123192
116	DISPOSITION_CHARGED_CLASS_3	0.052709
115	OFFENSE_CATEGORY_UUW - Unlawful Use of Weapon	0.046112
114	OFFENSE_CATEGORY_Homicide	0.043036

RF and XGB have different "top" features. Three feature lists will be created for further machine learning iterations:

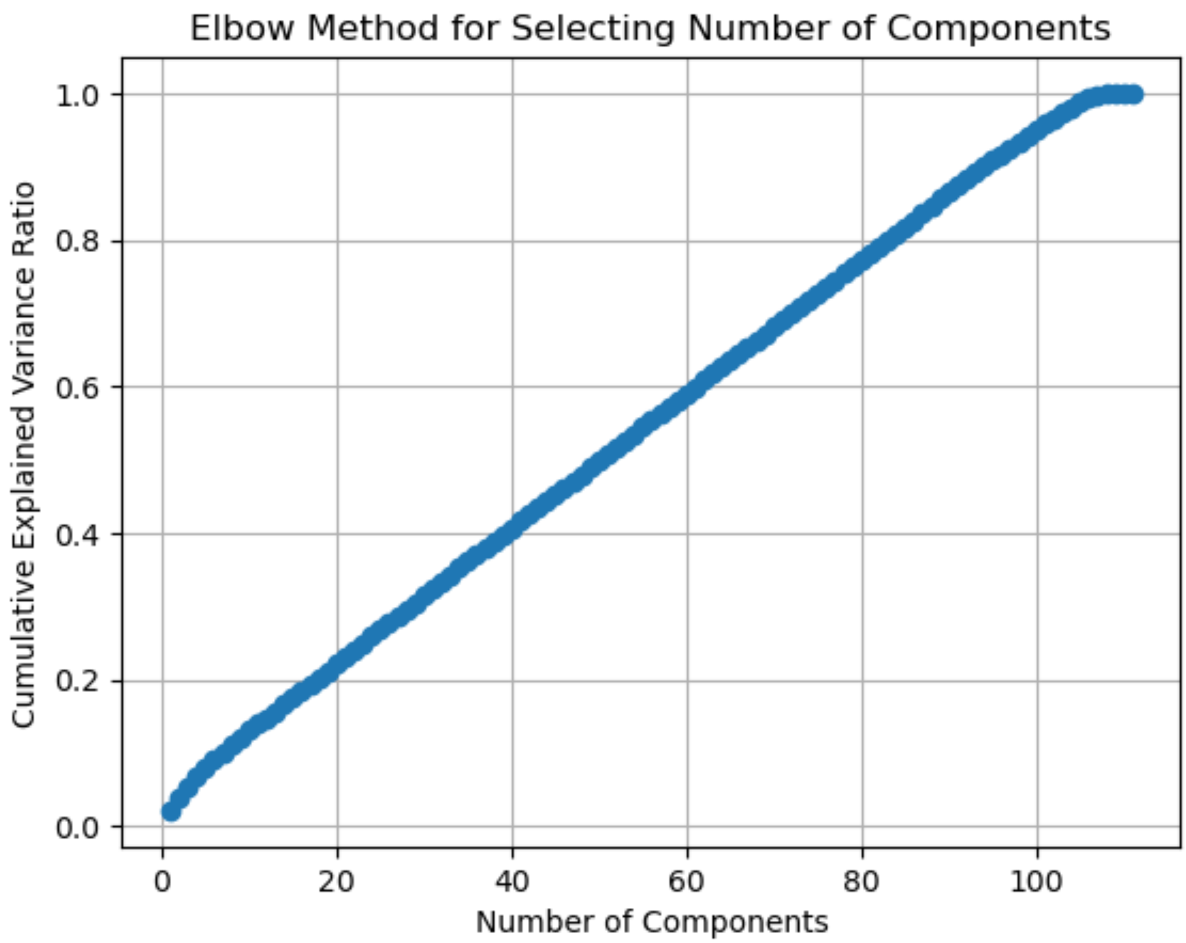
- RF top features will include the top 8 features and apply PCA to the remaining features.
- XGB top features will include the top 6 features and apply PCA to the remaining features.
- RF_XGB top features will include both of the top RF and XGB features and apply PCA to the remaining features.

To get the optimal number of components for the PCA, the elbow method will be used.

Top features for the RF model:

In [15]:

```
scaler = StandardScaler()
X_top_columns = data_dummy[rf_fi.sort_values(by = 'Importance', ascending =
X_rf = data_dummy[rf_fi.sort_values(by = 'Importance', ascending = False)[9:
X_rf_scaled = scaler.fit_transform(X_rf)

explained_variances = []
total_variances = []

for n_components in range(1, X_rf.shape[1]+1):
    pca = PCA(n_components=n_components)
    X_rf_pca = pca.fit_transform(X_rf_scaled)
    explained_variances.append(pca.explained_variance_ratio_)
    total_variances.append(sum(pca.explained_variance_ratio_))


plt.plot(range(1, X_rf.shape[1]+1), total_variances, marker='o', linestyle='
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('Elbow Method for Selecting Number of Components')
plt.grid(True)
plt.show()
```



In [16]:
```
index = next((i for i, var in enumerate(total_variances) if var > 0.75), Non
index
```

Out[16]: 78

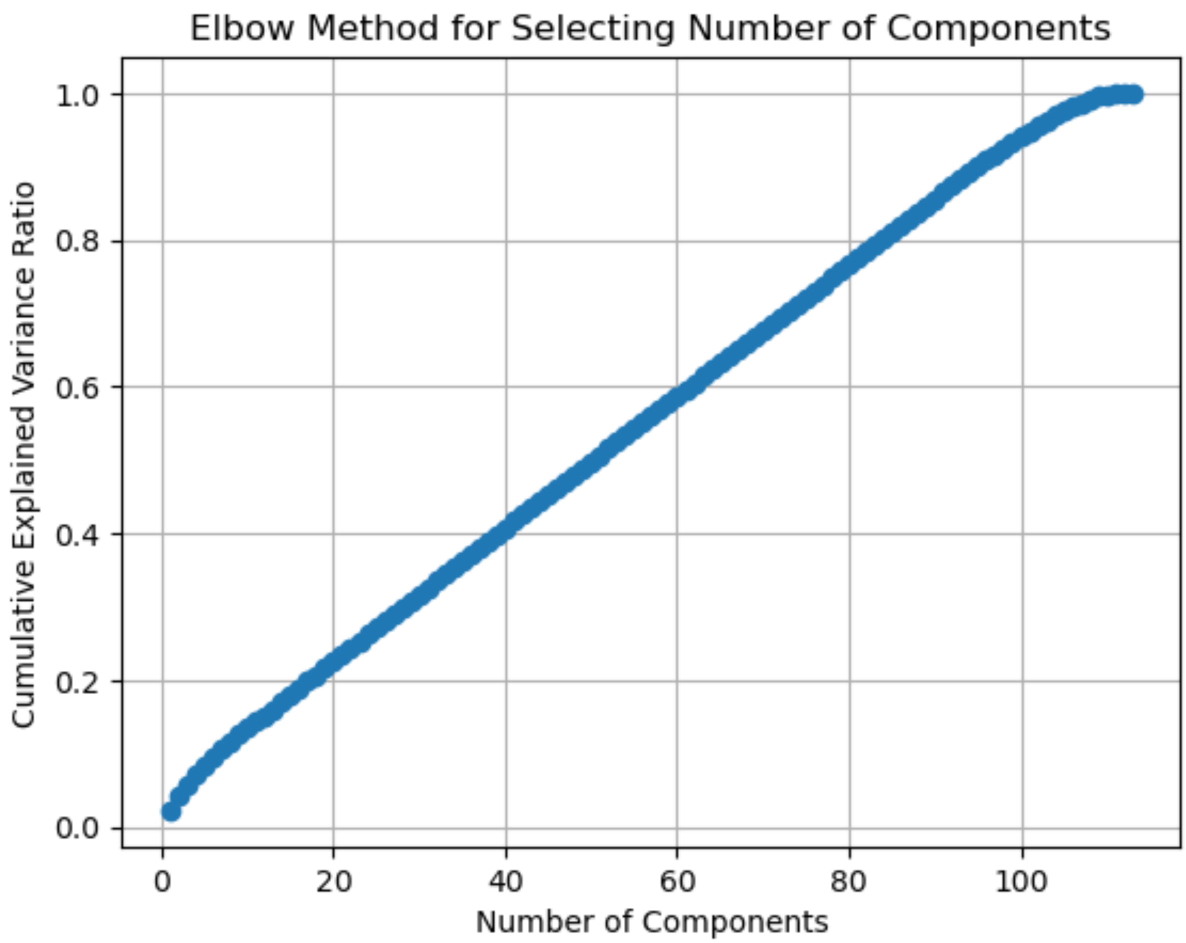
Looking at the explained variance plot, it appears that more components continue to add more explained variance. There does appear to be a small flat line at 105 components, but a reduction in 6 features does not seem worthwhile. To continue using PCA-applied variables, as there may still be noise reduction, computational efficiencies, and a reduction of overfitting, the number of components that explains 75% of the variance will be selected.

In [17]:
```
pca = PCA(n_components=78)
X_rf_pca = pca.fit_transform(X_rf_scaled)
print(pca.explained_variance_ratio_)
print('Total Explained Variance: '+ str(sum(pca.explained_variance_ratio_)))
X_rf_pca = pd.concat([pd.DataFrame(X_rf_pca), pd.DataFrame(X_top_columns)],
rf_pca_feature_list = rf_fi.sort_values(by = 'Importance', ascending = False
```

```
[0.02202295 0.01708048 0.01526366 0.01313667 0.01257351 0.01136296
 0.01063171 0.01053837 0.01038778 0.0103699  0.01016517 0.0098565
 0.00972361 0.00966915 0.00959702 0.00953726 0.00951215 0.0094469
 0.00941825 0.00939785 0.00935633 0.00931135 0.00929387 0.0092809
 0.00925161 0.00921362 0.00919352 0.00915726 0.00915035 0.00911954
 0.00910224 0.00908814 0.00908638 0.00907875 0.0090692  0.00906355
 0.00905493 0.0090509  0.00904967 0.00904208 0.00903852 0.00903398
 0.00903233 0.00902793 0.00902487 0.00902398 0.00902364 0.00902182
 0.00902088 0.00901804 0.0090177  0.00901635 0.00901583 0.00901551
 0.00901503 0.0090144  0.00901396 0.00901315 0.00901284 0.0090121
 0.00901148 0.00901124 0.00901107 0.00900955 0.00900305 0.00899422
 0.00898432 0.0089786  0.00896682 0.00895867 0.00893614 0.00892444
 0.00890001 0.00886679 0.00882881 0.00880779 0.00877453 0.0087404 ]
Total Explained Variance: 0.7538268124058978
```

Top features for the XGB model:

In [18]:
```
scaler = StandardScaler()
X_top_columns = data_dummy[xgb_fi.sort_values(by = 'Importance', ascending =
X_xgb = data_dummy[xgb_fi.sort_values(by = 'Importance', ascending = False)[
X_xgb_scaled = scaler.fit_transform(X_xgb)

explained_variances = []
total_variances = []

for n_components in range(1, X_xgb.shape[1]+1):
    pca = PCA(n_components=n_components)
    X_xgb_pca = pca.fit_transform(X_xgb_scaled)
    explained_variances.append(pca.explained_variance_ratio_)
    total_variances.append(sum(pca.explained_variance_ratio_))


plt.plot(range(1, X_xgb.shape[1]+1), total_variances, marker='o', linestyle=
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('Elbow Method for Selecting Number of Components')
plt.grid(True)
plt.show()
```



In [19]:
```
index = next((i for i, var in enumerate(total_variances) if var > 0.75), Non
index
```

Out[19]: 79

Based on the explained variance plot, it seems that more components contribute to additional explained variance. Although there's a slight flattening toward the end, similar to the curve of the RF features' explained variance, continuing with PCA-applied variables could still offer benefits such as noise reduction, computational efficiencies, and overfitting reduction. Therefore, the number of components explaining 75% of the variance will be chosen for further analysis.

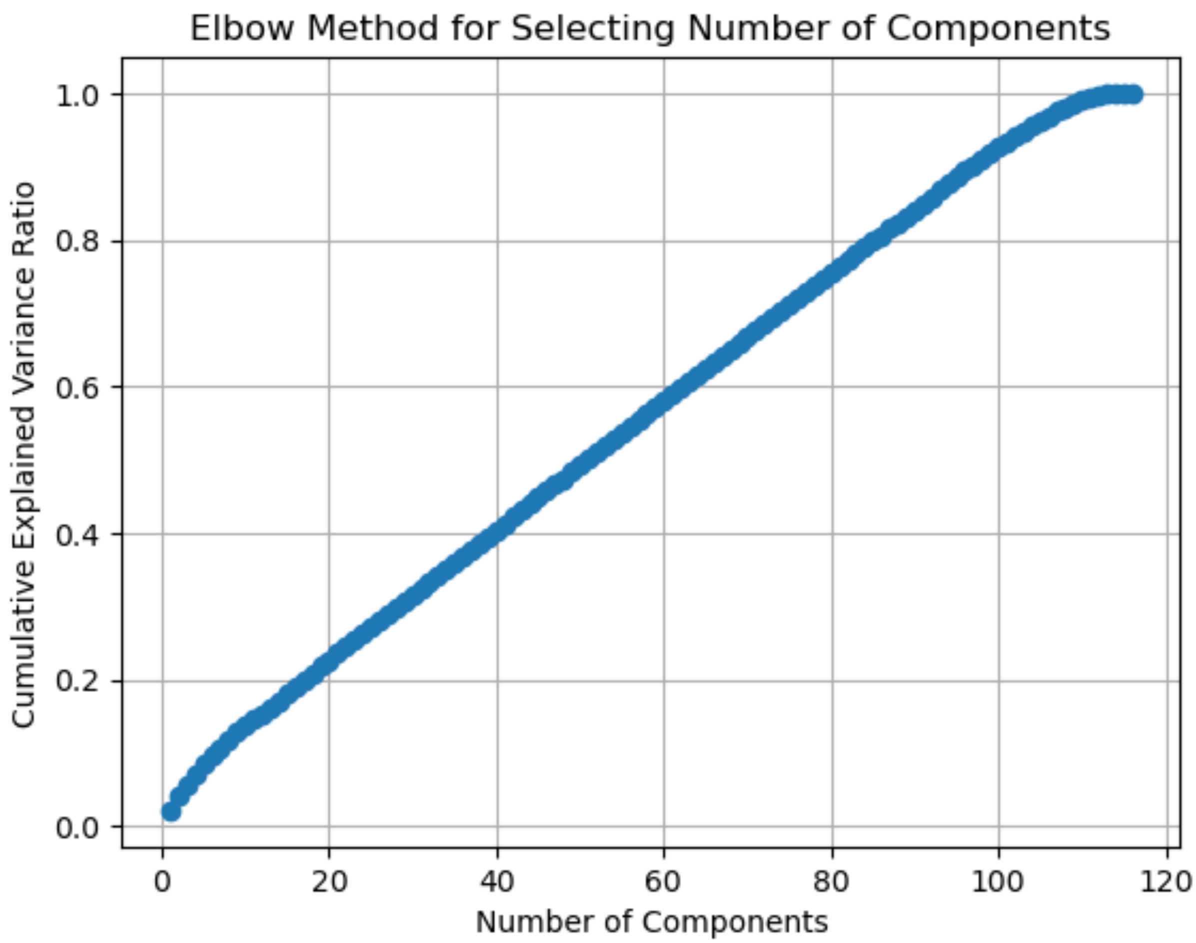
In [20]:
```
pca = PCA(n_components=79)
X_xgb_pca = pca.fit_transform(X_xgb_scaled)
print(pca.explained_variance_ratio_)
print('Total Explained Variance: '+ str(sum(pca.explained_variance_ratio_)))
X_xgb_pca = pd.concat([pd.DataFrame(X_xgb_pca), pd.DataFrame(X_top_columns)]
xgb_pca_feature_list = xgb_fi.sort_values(by = 'Importance', ascending = Fal
```

```
[0.02245162 0.01935735 0.01600801 0.01374007 0.01247135 0.01129108
 0.01089665 0.01065709 0.01043479 0.0103113  0.01014397 0.01001619
 0.00972264 0.00960915 0.00952526 0.00947519 0.00942009 0.00938422
 0.00932048 0.00927366 0.00925809 0.00922061 0.00918446 0.00913125
 0.00910332 0.00904978 0.00901102 0.00899741 0.00896786 0.00896608
 0.00894335 0.00893989 0.00892887 0.00891544 0.00890494 0.00889904
 0.00889625 0.00889176 0.0088862  0.00888312 0.00888177 0.00887447
 0.00887031 0.00886663 0.00886389 0.00886284 0.00886243 0.00886114
 0.00885898 0.0088588  0.00885799 0.0088569  0.0088564  0.00885583
 0.00885532 0.00885477 0.00885347 0.00885314 0.00885285 0.00885258
 0.00885061 0.00884995 0.00884519 0.00883627 0.00883027 0.00882692
 0.00882254 0.00880096 0.00879695 0.00879071 0.00876945 0.00875889
 0.00874611 0.00870753 0.0086766  0.00864792 0.00864296 0.00858442
 0.0085613 ]
Total Explained Variance: 0.7570449877623912
```

The top features using both RF and XGB:

In [21]:

```
scaler = StandardScaler()
X_top_columns = data_dummy[list(set(rf_fi.sort_values(by = 'Importance', asc
X_combo = data_dummy[list(set(rf_fi.sort_values(by = 'Importance', ascending
X_combo_scaled = scaler.fit_transform(X_combo)

explained_variances = []
total_variances = []

for n_components in range(1, X_combo.shape[1]+1):
    pca = PCA(n_components=n_components)
    X_combo_pca = pca.fit_transform(X_combo_scaled)
    explained_variances.append(pca.explained_variance_ratio_)
    total_variances.append(sum(pca.explained_variance_ratio_))


plt.plot(range(1, X_combo.shape[1]+1), total_variances, marker='o', linestyl
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('Elbow Method for Selecting Number of Components')
plt.grid(True)
plt.show()
```



In [22]:
```
index = next((i for i, var in enumerate(total_variances) if var > 0.75), Non
index
```

Out[22]:
```
80
```

The same pattern is exhibited with the explained variance plot taking the top features from both the RF and XGB model. As with before, the component number that first crosses 75% explained variance will be selected.

In [23]:
```
pca = PCA(n_components=80)
X_combo_pca = pca.fit_transform(X_combo_scaled)
print(pca.explained_variance_ratio_)
print('Total Explained Variance: '+ str(sum(pca.explained_variance_ratio_)))
X_combo_pca = pd.concat([pd.DataFrame(X_combo_pca), pd.DataFrame(X_top_colum
combo_pca_feature_list = list(set(rf_fi.sort_values(by = 'Importance', ascen
list(set(rf_fi.sort_values(by = 'Importance', ascending = False)[:8]['Featur
```

```
[0.0219975  0.02012931 0.01588217 0.01396167 0.01324855 0.01220155
 0.01129928 0.01073961 0.01039697 0.01019764 0.01011832 0.00998821
 0.0098114  0.00957513 0.00939867 0.00935859 0.00929507 0.00917398
 0.00913597 0.0091339  0.00900093 0.00898008 0.00895307 0.00891327
 0.00890133 0.00886898 0.00883803 0.00881963 0.00878431 0.00874905
 0.00874597 0.008728   0.00871858 0.00870968 0.00869315 0.00869073
 0.00868301 0.00867079 0.00866391 0.00865881 0.00865539 0.00864854
 0.00864475 0.0086431  0.00864188 0.00863781 0.00863637 0.00863586
 0.00863288 0.00863166 0.00863092 0.00863018 0.00862938 0.00862781
 0.00862698 0.00862641 0.00862576 0.00862502 0.00862441 0.00862424
 0.00862368 0.00862344 0.00862097 0.00861415 0.0086111  0.00860844
 0.00859726 0.00858866 0.00858459 0.00857656 0.00857035 0.00856526
 0.0085422  0.00851561 0.00849368 0.00847667 0.00844464 0.00841762
 0.00840057 0.0083707 ]
Total Explained Variance: 0.754340259326382
```

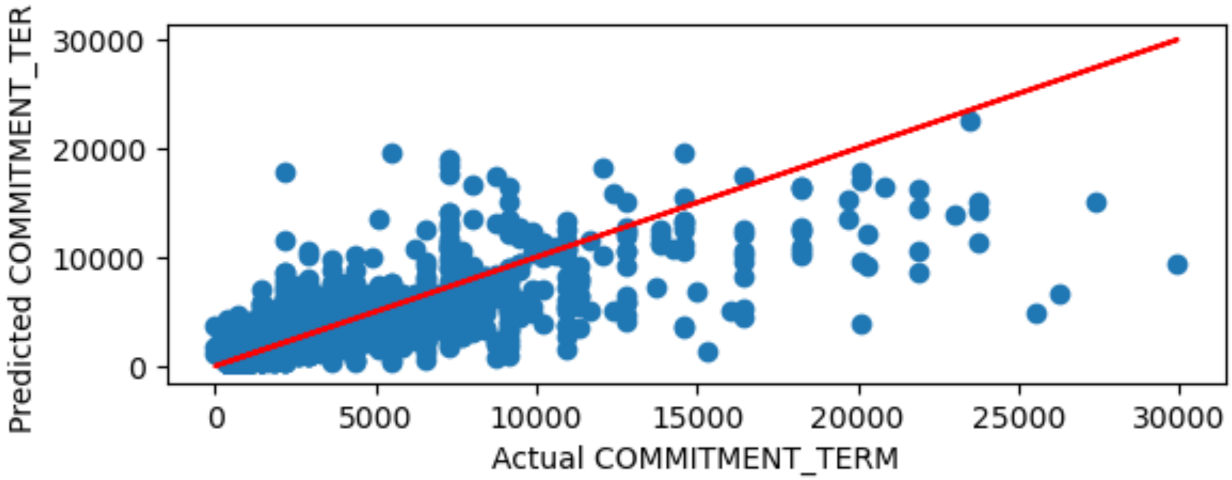
Machine Learning: RF with PCA applied data sets

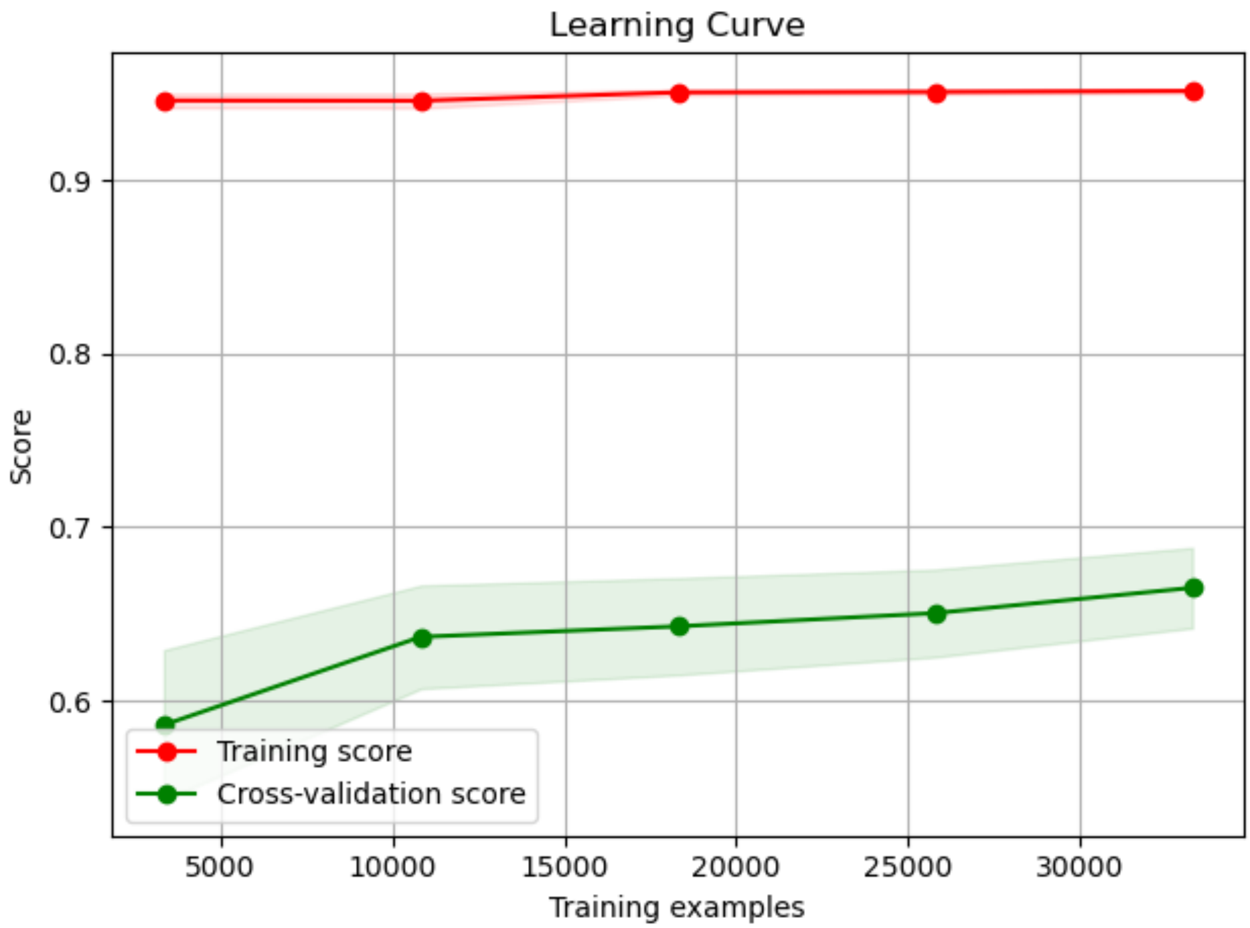
- Fit RF and XGB with default parameters using PCA applied data sets

In [615…

```
X = X_rf_pca
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
rf2 = RandomForestRegressor(n_jobs = -1, random_state=0)
rf2.fit(X_train, y_train)

# Predictions
plm = rf2.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(rf2, X_train, y_train, cv=5, n_jobs = -1, scorin
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(rf2, "Learning Curve", X_train, y_train, cv=5, n_jobs=-1

plt.tight_layout()
plt.show()
```

```
MSE: 1138614.586815579
RMSE: 1067.0588488061842
Relative RMSE: 0.6627326375891726
MAE: 510.92364055127683
Relative MAE: 0.31732623958654116
EV: 0.6836535718008591
R2: 0.6826123596344158
Cross-Validation RMSE: 1074.9838084201353
```

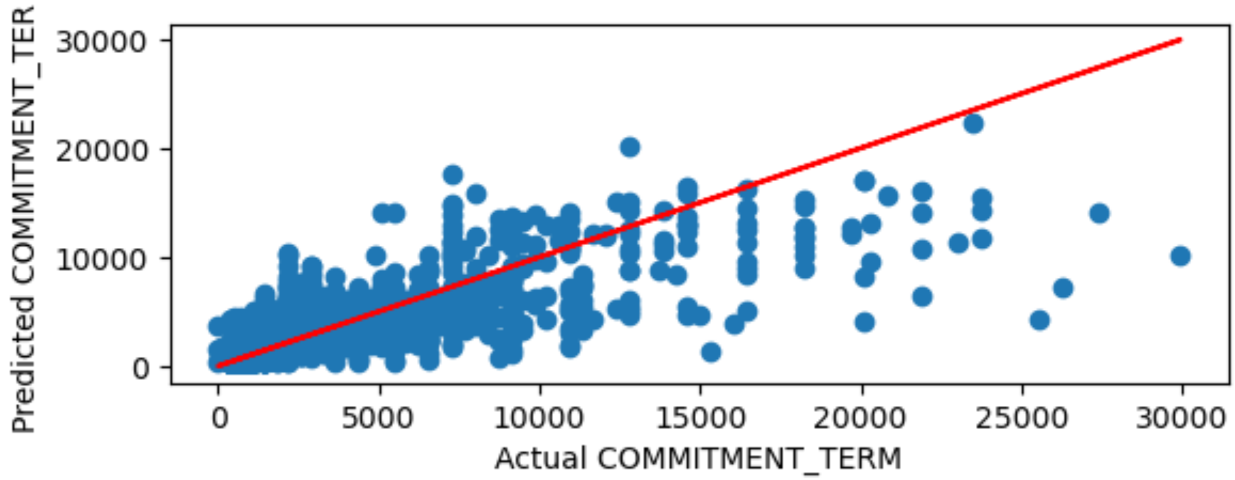

Predicted COMMITMENT_TER
Actual COMMITMENT_TERM
30000
20000
10000
0
0
5000
10000
15000
20000
25000
30000


Learning Curve
Score
Training examples
0.9
0.8
0.7
0.6
5000
10000
15000
20000
25000
30000
Training score
Cross-validation score

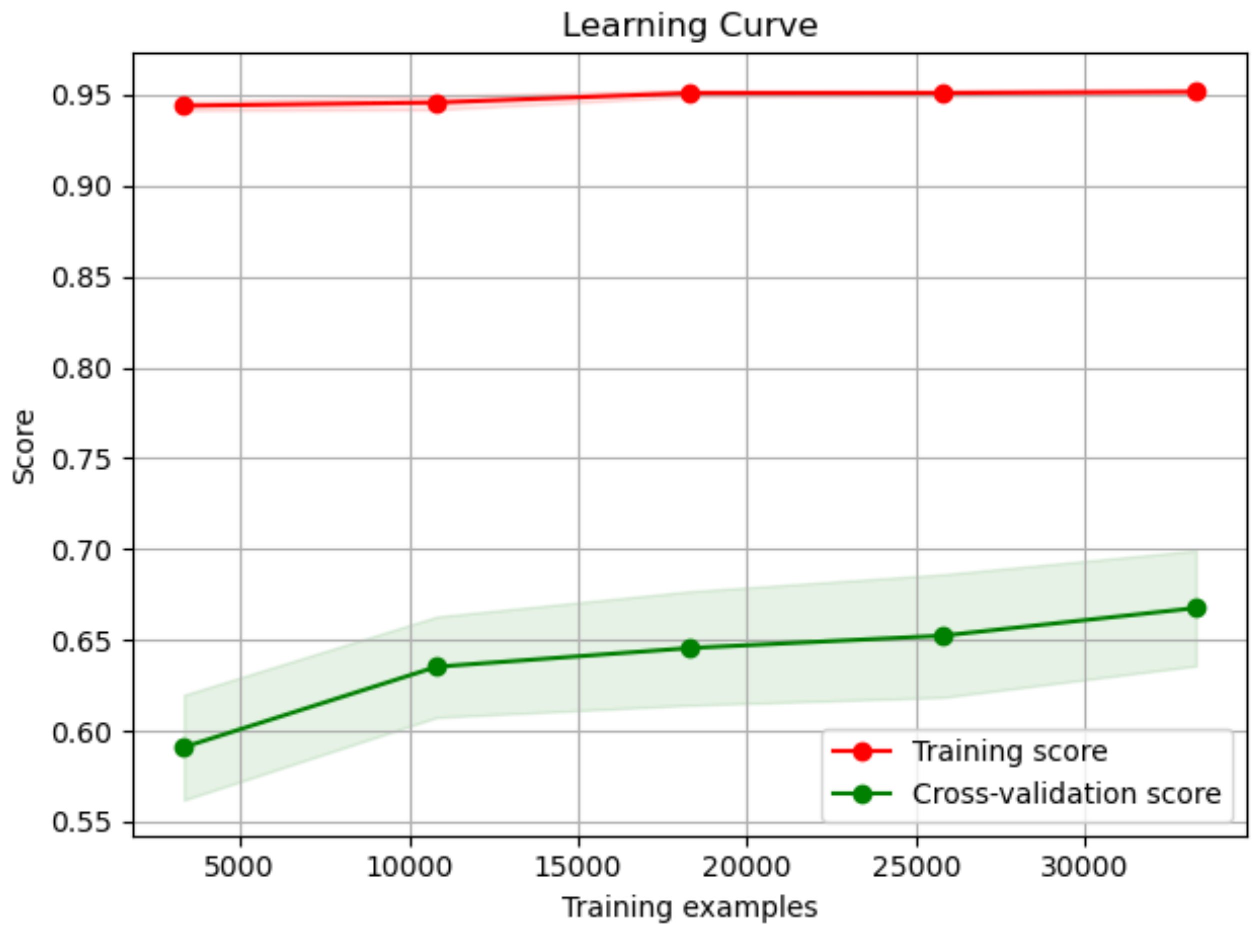
In [616…

```
X = X_xgb_pca
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
rf3 = RandomForestRegressor(n_jobs = -1, random_state=0)
rf3.fit(X_train, y_train)

# Predictions
plm = rf3.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(rf3, X_train, y_train, cv=5,n_jobs = -1, scoring
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(rf3, "Learning Curve", X_train, y_train, cv=5, n_jobs=-1

plt.tight_layout()
plt.show()
```

```
MSE: 1094755.0763643503
RMSE: 1046.30544123805
Relative RMSE: 0.6498430387146771
MAE: 509.46773721890384
Relative MAE: 0.3164220020586692
EV: 0.6957468820157247
R2: 0.694838152884296
Cross-Validation RMSE: 1071.6305236322403
```



30000
20000
10000
0
Predicted COMMITMENT_TER
0
5000
10000
15000
20000
25000
30000
Actual COMMITMENT_TERM

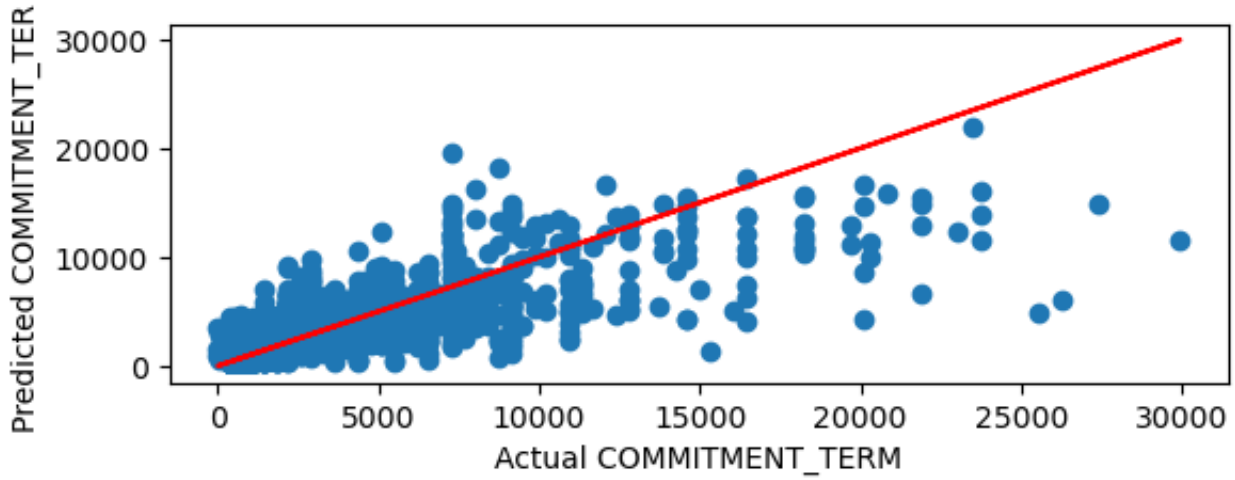


Learning Curve
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
Score
5000
10000
15000
20000
25000
30000
Training examples
Training score
Cross-validation score

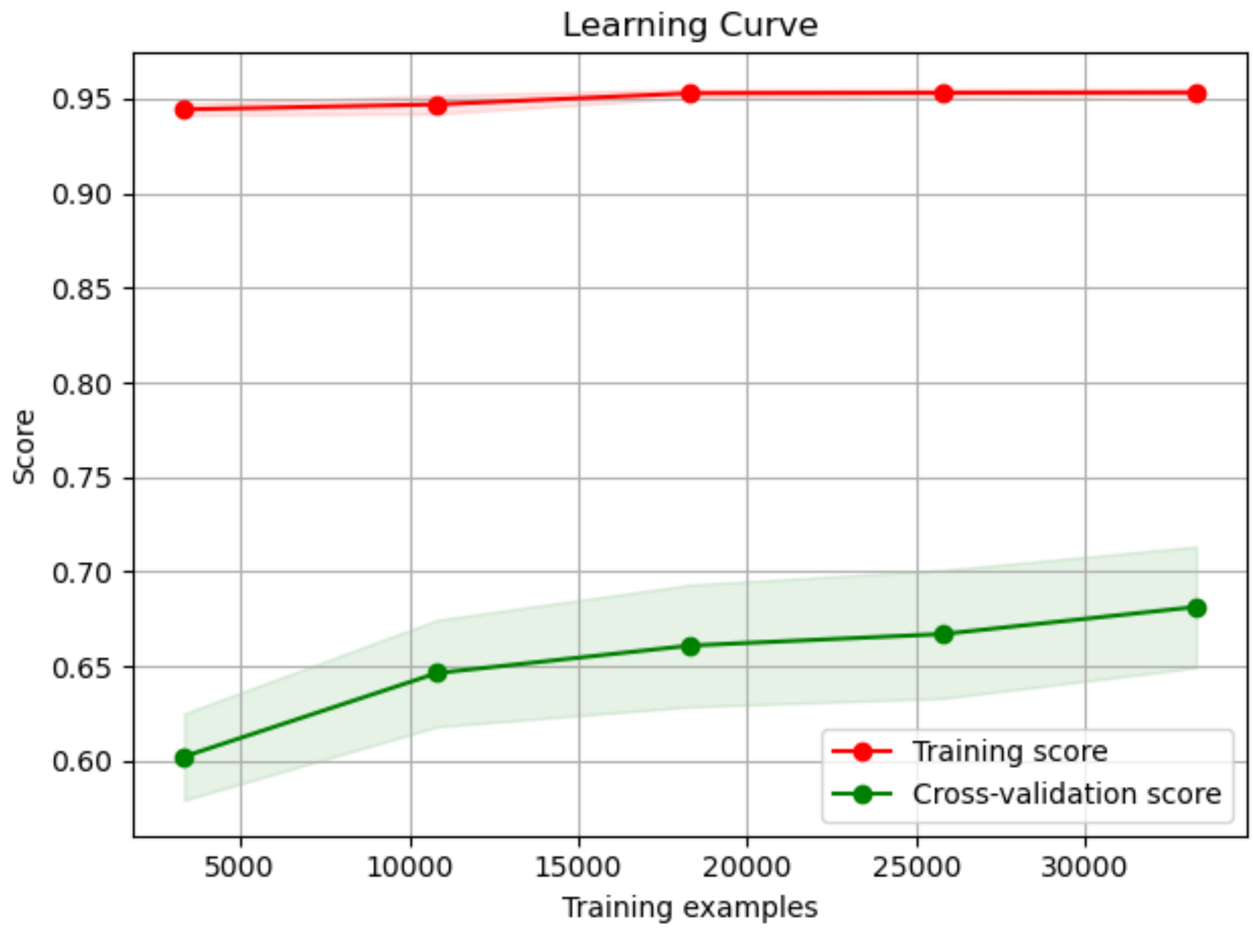
In [24]:

```
X = X_combo_pca
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
rf4 = RandomForestRegressor(n_jobs = -1, random_state=0)
rf4.fit(X_train, y_train)

# Predictions
plm = rf4.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(rf4, X_train, y_train, cv=5,n_jobs = -1, scoring
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(rf4, "Learning Curve", X_train, y_train, cv=5, n_jobs=-1

plt.tight_layout()
plt.show()
```

```
MSE: 1069073.4558393748
RMSE: 1033.960084258273
Relative RMSE: 0.6421755412731431
MAE: 502.304269207302
Relative MAE: 0.311972890320431
EV: 0.7030106504805198
R2: 0.7019968780873367
Cross-Validation RMSE: 1047.6902973405968
```



Predicted COMMITMENT_TER
30000
20000
10000
0
0
5000
10000
15000
20000
25000
30000
Actual COMMITMENT_TERM



Learning Curve
Score
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
5000
10000
15000
20000
25000
30000
Training examples
Training score
Cross-validation score

The RF model fitted with the PCA-applied datasets using the top RF features (RF2) demonstrates underperformance across all metrics compared to the baseline RF model. Examination of the CV RMSE and the learning curve suggests potential underfitting, possibly due to excessive dimensionality reduction. Although the CV R2 score appears to converge with the training R2 score, additional training examples or a more complex model with optimized hyperparameters may be necessary. While the CV shaded area tightens with continued training, indicating a reduction in variance, the complexity introduced by PCA in RF2 does not seem warranted given the reasonably good performance of the baseline RF model without significant overfitting.

Similarly, the RF model fitted with the PCA-applied datasets using the top XGB features (RF3) performs similarly across all metrics compared to the baseline RF model. However, the divergence between the CV RMSE and the RMSE is more pronounced in RF3. The training curve fails to converge, and the training R2 remains notably higher than the CV R2 score. The lack of tightening in the CV shaded area with continued training suggests that the model fails to reduce variance effectively. Consequently, the additional steps taken with PCA-applied variables in RF3 do not seem justified.

The RF model fitted with the PCA-applied datasets using the top XGB and RF features (RF4) also exhibits similar performance across all metrics compared to the baseline RF model. However, akin to RF3, the CV RMSE diverges further from the RMSE, and the training curve fails to converge. The training R2 remains significantly higher than the CV R2 score, indicating potential overfitting. As with RF2 and RF3, the inclusion of PCA-applied variables in RF4 does not seem warranted.

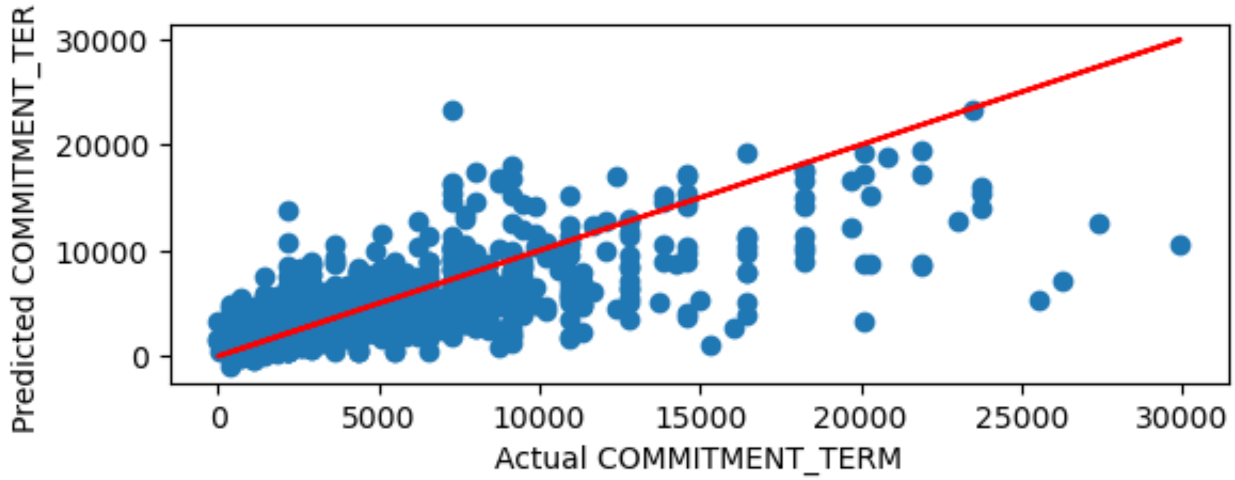
Machine Learning: XGB with PCA applied data sets

- Fit RF and XGB with default parameters using PCA applied data sets

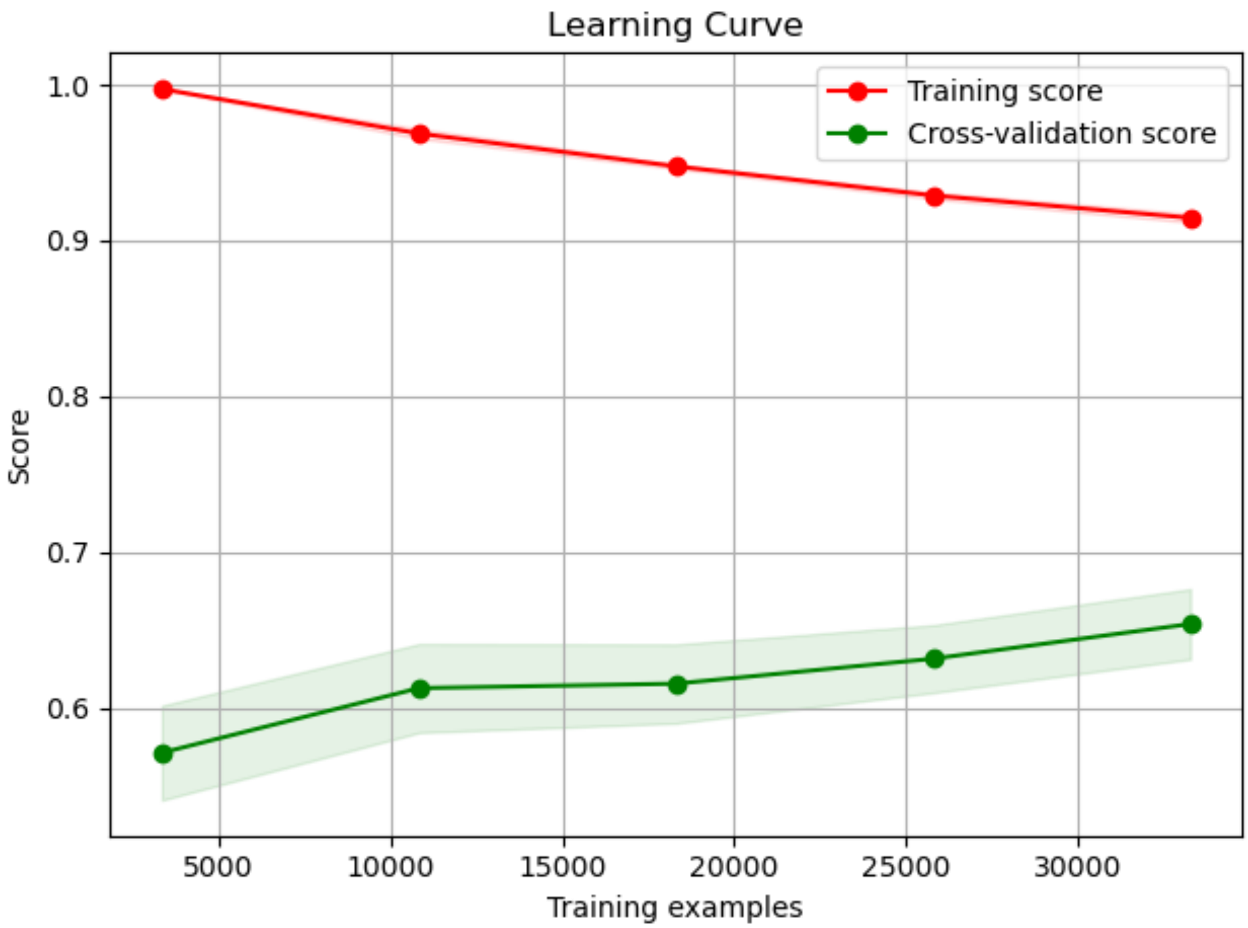
In [25]:

```
X = X_rf_pca
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
xgb2 = XGBRegressor(n_jobs = -1, random_state=0)
xgb2.fit(X_train, y_train)

# Predictions
plm = xgb2.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(xgb2, X_train, y_train, cv=5, n_jobs = -1, scori
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(xgb2, "Learning Curve", X_train, y_train, cv=5, n_jobs=-

plt.tight_layout()
plt.show()
```

```
MSE: 1169831.5488915923
RMSE: 1081.587513283873
Relative RMSE: 0.6717561512788994
MAE: 521.6742182095708
Relative MAE: 0.324003245915723
EV: 0.6739301424998339
R2: 0.6739106636896995
Cross-Validation RMSE: 1091.5491668402126
```



Predicted COMMITMENT_TER
Actual COMMITMENT_TERM

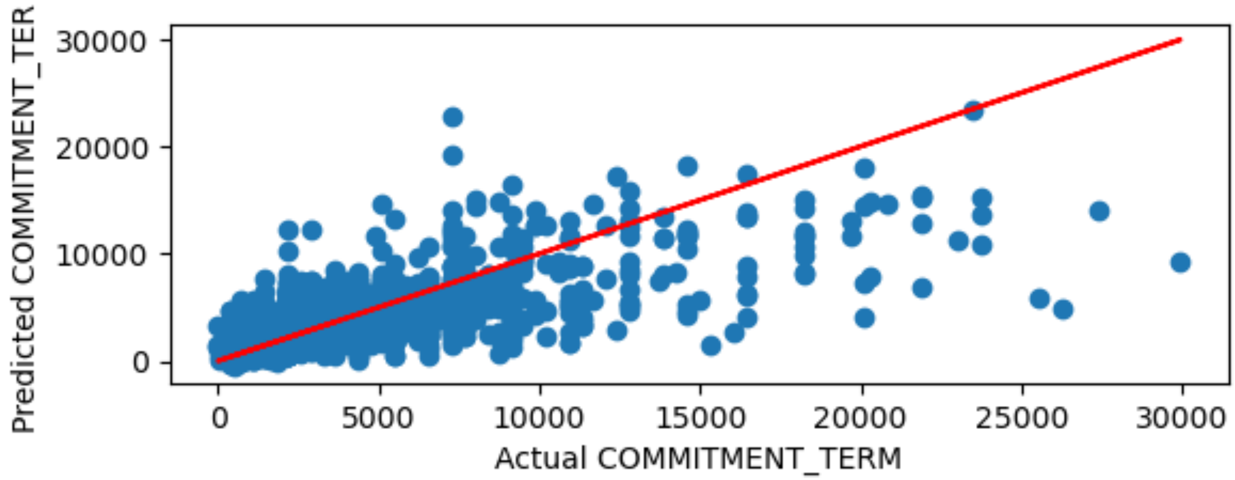


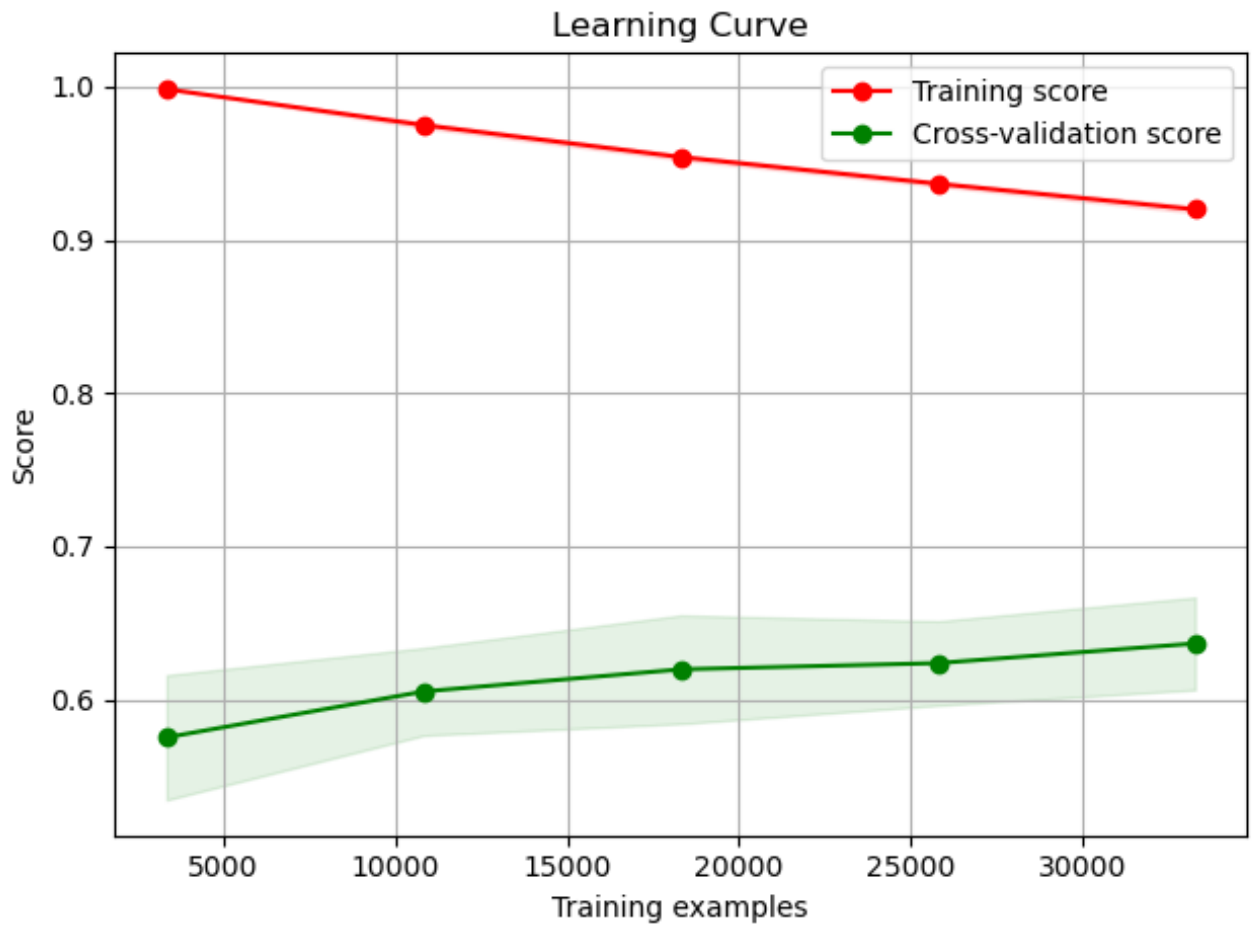
Learning Curve
Training score
Cross-validation score
Score
Training examples

In [26]:

```
X = X_xgb_pca
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
xgb3 = XGBRegressor(n_jobs = -1, random_state=0)
xgb3.fit(X_train, y_train)

# Predictions
plm = xgb3.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(xgb3, X_train, y_train, cv=5, n_jobs = -1, scori
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(xgb3, "Learning Curve", X_train, y_train, cv=5, n_jobs=-

plt.tight_layout()
plt.show()
```

```
MSE: 1221888.0387162834
RMSE: 1105.3904462751084
Relative RMSE: 0.686539760056699
MAE: 534.3287382526141
Relative MAE: 0.33186275943265225
EV: 0.6594032729692303
R2: 0.6593999709034934
Cross-Validation RMSE: 1119.490411132189
```

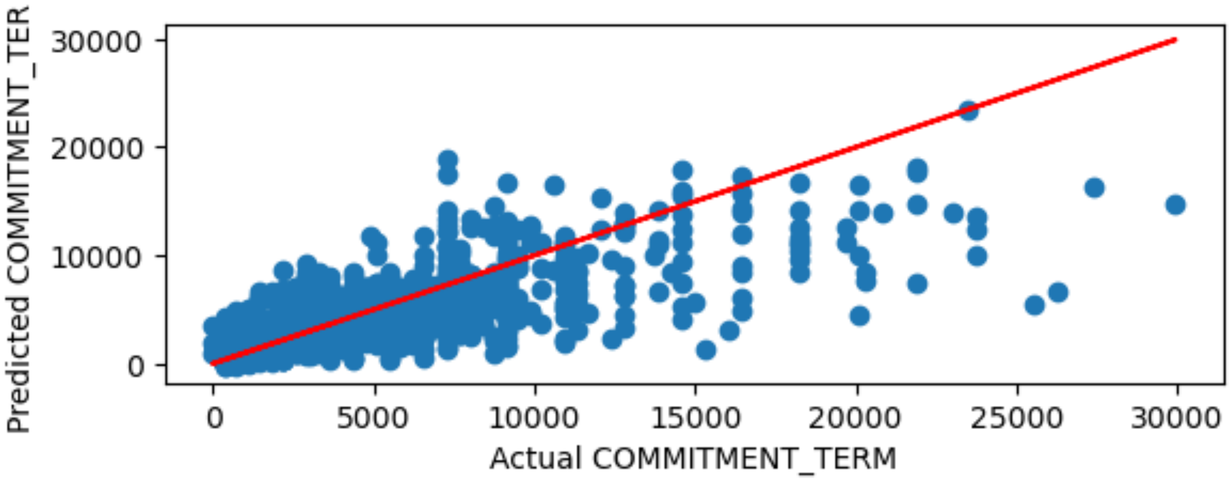

Predicted COMMITMENT_TER
Actual COMMITMENT_TERM
30000
20000
10000
0
0
5000
10000
15000
20000
25000
30000

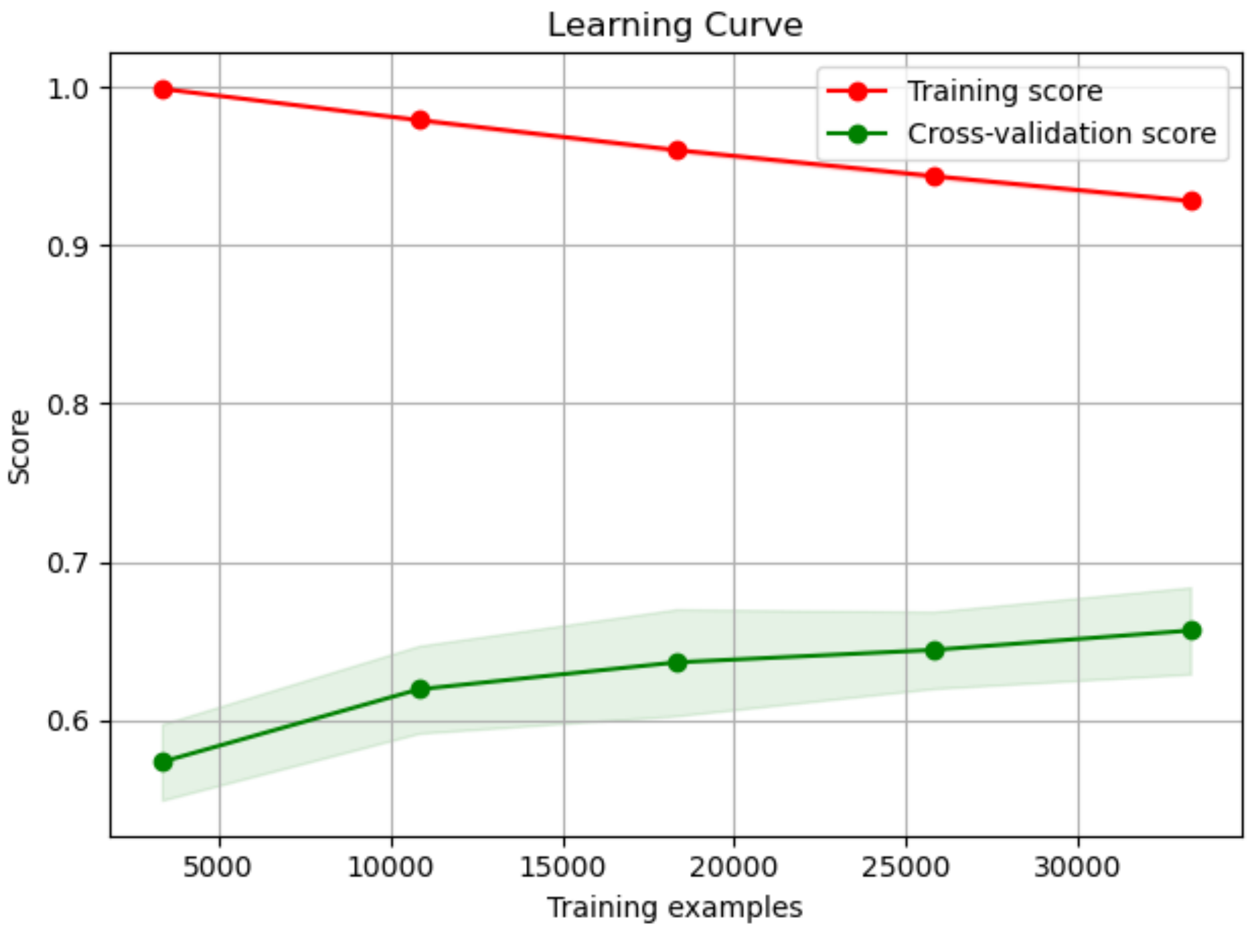

Learning Curve
Training score
Cross-validation score
Score
Training examples
1.0
0.9
0.8
0.7
0.6
5000
10000
15000
20000
25000
30000

In [27]:

```
X = X_combo_pca
y = data_dummy[col].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra

# Regression modeling
xgb4 = XGBRegressor(n_jobs = -1, random_state=0)
xgb4.fit(X_train, y_train)

# Predictions
plm = xgb4.predict(X_test)

# Plotting Actual vs. Predicted values
plt.subplot(2, 1, 2)
plt.scatter(y_test, plm)
plt.plot(y_test, y_test, "r")
plt.xlabel('Actual ' + col)
plt.ylabel('Predicted ' + col)

# Evaluate the model
print("MSE:", metrics.mean_squared_error(y_test, plm))
print("RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)))
print("Relative RMSE:", np.sqrt(metrics.mean_squared_error(y_test, plm)) / y
print("MAE:", metrics.mean_absolute_error(y_test, plm))
print("Relative MAE:", metrics.mean_absolute_error(y_test, plm) / y_test.mea
print("EV:", metrics.explained_variance_score(y_test, plm))
print("R2:", metrics.r2_score(y_test, plm))

# Validation Curve Plot
cv_scores = cross_val_score(xgb4, X_train, y_train, cv=5, n_jobs = -1, scori
print("Cross-Validation RMSE:", np.sqrt(-cv_scores.mean()))
plot_learning_curve(xgb4, "Learning Curve", X_train, y_train, cv=5, n_jobs=-

plt.tight_layout()
plt.show()
```

```
MSE: 1115496.1457149836
RMSE: 1056.1705097733904
Relative RMSE: 0.6559700699442467
MAE: 518.5980499243643
Relative MAE: 0.32209268857054574
EV: 0.6890595364979086
R2: 0.6890566012195951
Cross-Validation RMSE: 1086.8741767213887
```





The XGB model fitted with the PCA-applied datasets using the top RF features (XGB2) demonstrates underperformance across all metrics compared to the baseline XGB model. Analysis of the CV RMSE and the learning curve suggests potential underfitting, with the CV R2 score converging with the training R2 score to a lesser extent than the baseline XGB model. While the CV shaded area tightens with continued training, indicating some reduction in variance, the complexity introduced by PCA in XGB2 does not appear justified given the reasonably good performance of the baseline XGB model without significant overfitting.

Similarly, the XGB model fitted with the PCA-applied datasets using the top XGB features (XGB3) also underperforms across all metrics compared to the baseline XGB model. The divergence between the CV RMSE and the RMSE is more pronounced in XGB3, with the training curve slowly converging but the gap remaining larger than in the baseline XGB model. The lack of tightening in the CV shaded area with continued training suggests that the model fails to effectively reduce variance. Consequently, the additional steps taken with PCA-applied variables in XGB3 do not seem justified.

Likewise, the XGB model fitted with the PCA-applied datasets using the top XGB and RF features (XGB4) exhibits similar performance across all metrics compared to the baseline XGB model. However, as with XGB3, the CV RMSE diverges further from the RMSE, and the training curve fails to converge, with the training R2 remaining significantly higher than the CV R2 score. As with XGB2 and XGB3, the inclusion of PCA-applied variables in XGB4 does not seem warranted.

Summary: Model Metrics

The R2, RMSE (and CV RMSE), and MAE metrics were used to analyze and evaluate the regressor models.

Each metric captures a different aspect of model performance. RMSE measures the average deviation of predicted values from actual values, but penalizes larger errors more heavily due to squaring. MAE measures the average deviation but does not penalize larger errors, unlike the RMSE. RMSE and MAE are both in the same unit as the predicted variable, terms sentenced, making them easy to interpret.

All models had a higher RMSE score than an MAE score. The MAE on a relative scale was around 30%, while the relative RMSE was closer to 60%. This can be interpreted as the models were mostly predicting with a relatively moderate margin of error, but occasionally missing large on some predictions.

The R^2 measures how well the model explains the variability in the predicted value. By considering all three metrics together, there is a much more comprehensive understanding of a regressor model's performance.

Summary: Model Performance

The benchmark linear model fitted with no multicollinear variables performed quite well. The R2 score remained within 8.2% of the highest R2 score from all iterations of models. The RMSE metric and the MAE metrics were both within 7.7% and 15.58% of the lowest RMSE and MAE scores, respectively. Due to the lack of multicollinearity and overfitting in the linear model, the ridge regularization did not provide significant improvement over the linear model baseline. The largest improvement was a reduction in the gap between CV RMSE and RMSE, a -0.7% reduction.

The benchmark RF model performed better across all metrics compared to the baseline linear model. However, the benchmark RF model showed signs of overfitting to the training model and there was room for more training examples to reduce variation in the cross-validated R2 scores. The benchmark RF model, however, had the tightest CV RMSE - RMSE gap out of all iterations of models and the lowest MAE.

All three of the RF models trained with PCA-applied datasets failed to perform significantly better than the benchmark RF model. RF2, which was trained using the top features identified using the XGB benchmark model with PCA applied to reduce the rest of the features down to 79 components, showed potential for improvement with more training. The variance seemed to be reducing with more training sessions, and the training and cross-validated R2 scores were moving toward convergence, although there was still a gap. RF3 and RF4 both saw reductions in the RMSE and MAE with improvements in the R2, however, also saw the CV RMSE gap increase drastically compared with the RF benchmark.

The benchmark XGB Model performed quite well, holistically, out of all benchmark models. The XGB benchmark model scored higher across all metrics compared to the linear model, and plotted a much healthier looking learning curve, with a smaller gap, than the random forest benchmark model. The training curve R2 was decreasing and converging with an increasing CV R2, and the variance of the model seemed to be reducing over training sessions as well. The XGB model has room for further training or hyperparameter tuning to close the gap.

All three XGB models trained with PCA-applied datasets had an increase in the CV RMSE gap compared to the benchmark XGB model. Additionally, all three models showed signs of converging training and CV R2 scores. Only XGB4 saw an improvement in the RMSE and R2 metric, however also saw the largest increase in the CV RMSE gap as well as the lowest change in model variance with training. Looking at the learning curves, XGB2 and XGB3 both saw decreases in model variance as training increased, XGB2 showing slightly tighter CV shaded areas.

Discussion and Conclusion

In conclusion, I would pick the XGB benchmark model as the overall best choice for this problem of predicting original term lengths for prison sentencing in Cook County. While in the middle of the pack in terms of RMSE, MAE, and R2 scoring, the learning curves showed the best ability for improvement using improved hyperparameter tuning and further training. Another strong option would be the linear regression benchmark model, which performed quite well in comparison to the more complex RF and XGB models. While the variance in the model appears quite high, potentially the largest CV shaded area across all models, the simplicity of the model and interpretability of the model makes it a compelling choice considering the metrics were not too far off the more complex models.

As with predictions of most human judgment and decision making, developing a model that can capture the complexity behind the human thinking pattern is quite difficult. Even with a topic such as sentencing term lengths, which have standard guidelines for the judge to follow, the case complexity, background of the participant, or information regarding the judge all play a huge role in the sentencing decision. All of those variables were not available from the original dataset and would require extensive research and data enrichment.

For future iterations of the project, dropping down to a specific court or judge could minimize some of the complex variations that the data has no features to explain. Another consideration for improvement would be to leverage GridSearchCV and fine-tune the hyperparameters of the model. This would require strong computational power as the data set is quite complex with a decent amount of rows. Running the models with CV was often too much compute for my laptop, so the GridSearchCV was skipped for this project.

Some hyperparameters to consider would be:

- XGB:
  - learning_rate: lower values make the module more robust but may require more boosting
  - n_estimators: the number of boosting or trees to build. Decreasing this figure would reduce overfitting risks
  - max_depth: the deeper the trees are, the more information and complex relations it can capture, with the risk of more overfitting
- RF
  - n_estimators: the number of trees in the forest. Increasing this will require more computation.
  - min_samples_split: the minimum sample to make a node. Increasing this will reduce overfitting risks
  - max_features: the number of features when looking for the best split

```
In [ ]:
```