Crime Data Report

By Dylan Sheldon

University of Gloucestershire

BSc Artificial Intelligence

Contents

Data Preparation

Two import methods had to be used as the layout of the excel files changed after January to March of 2021. The excel file names and sheet names had to be modified so that they were able to be opened in python. The older layout sheets were given a 0 in front of the file name and 1 for the latter. Some sheets were renamed P1a to follow the naming convention of the majority. The final manual adjustment was to insert 1 row below line 2 in the data of April to June 2021 because of the text difference at the top of the sheet. The newer layout sheets were simple to import but the older layout had extra blank columns and rows that were added to visually improve the excel file but made it difficult when importing it. Columns D and W, and rows 6 and 4 were blank and were removed in code. The total crime recorded column had to be relocated 1 row down as it was linked to row 4 which was to be dropped. Rows 1 to 3 and 62 onwards of the old format were ignored and so was rows 1 to 6 and 63 onwards as they contained notes and headers.

June 2019 to March 2020 does not contain a 3-month breakdown but only a rolling total to date. Therefore, it would be suitable to use the data from April 2020 onwards where that breakdown is available so insights into the data can be drawn for those periods which resulted in data for 24 months. The feature that was dropped was the area code as it related to the area name but doesn't provide any meaningful information when compared against any of the features of the dataset and the area name is simpler to understand. The excel data contained bold columns that signify the category of crime which is the sum of its sub-categories. Such as violence against the person being the sum of homicide, violence with and without injury, stalking and harassment, and death or serious injury – unlawful driving. The other main categories are sexual offences, robbery, theft, criminal damages, drugs, possession of weapons, public orders, and miscellaneous crimes. Due to the quantity crimes, three sub-categorical crimes from two different categories were selected. These were: violence with injury and stalking and harassment from violence against the person; and burglary from theft offences. The data contained immediate outliers which were identified to be the total crimes for England and Wales, city of London, and the Metropolitan police. Apart from the city of London, the datapoints were significantly larger than the other areas and the city of London contained null values, so those records were dropped. It was clear to see when identifying outliers that there were 2 possible divisions that could be made within the data as it contained the area names for electoral regions and counties. As the electoral regions are the sum of the counties, the county data will be carried forward and as there are more data points available than the regions. Based on the date range named in the excel file, a date-time column was added to each data frame in code before merging. The date took the last day of the specified month as the next file continued from the first of the month after. By adding a date column, it allows the data to be sorted and filtered once merged as there were no other identifiers that could have been used. The data was then cleaned after the z-score for each column had been calculated and scores which exceeded 3 in all columns were dropped.

Dataset Analysis

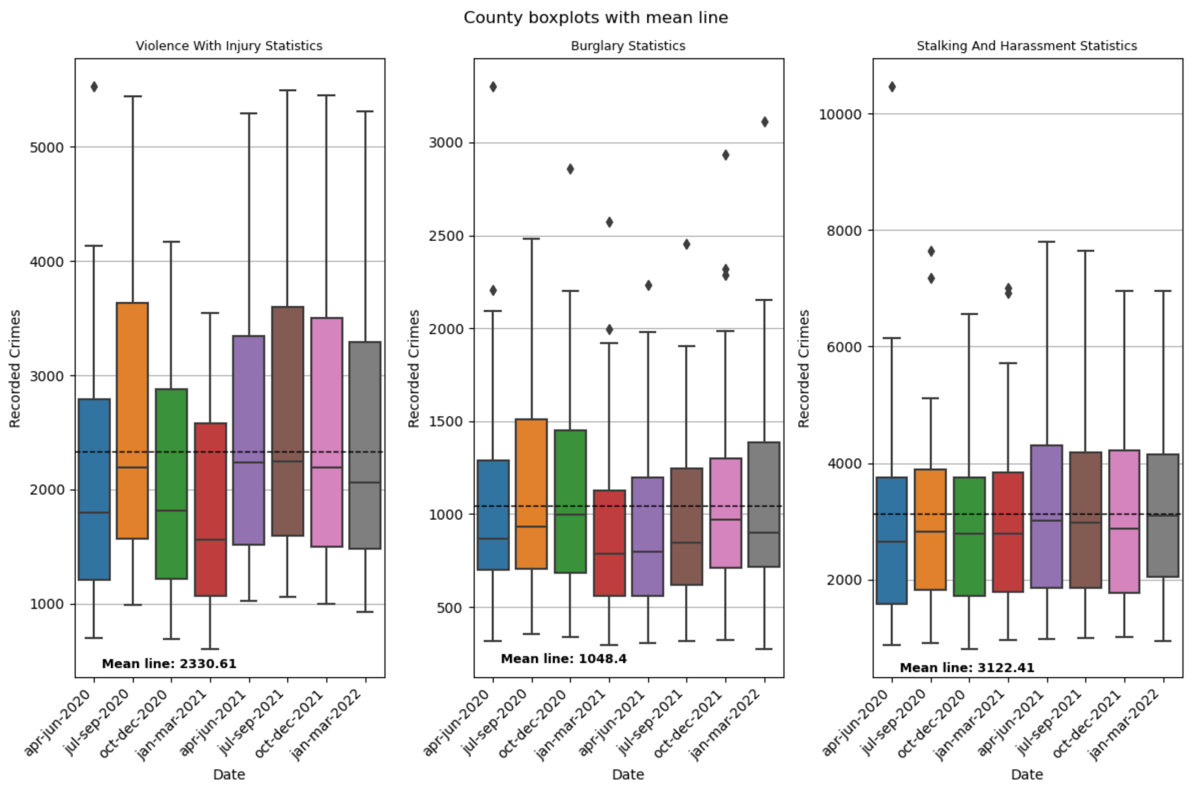
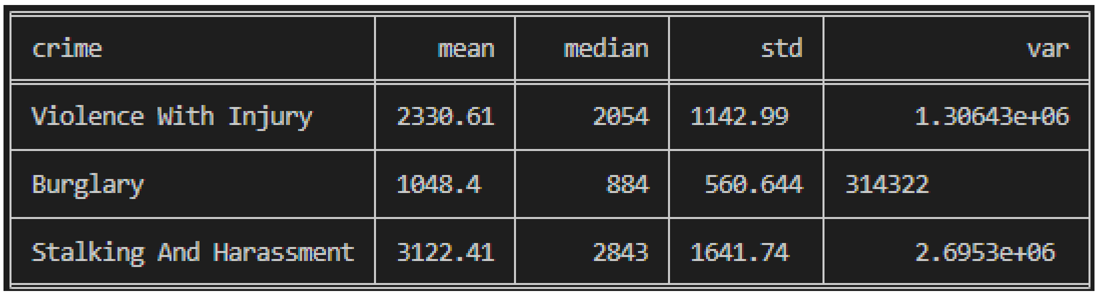


Figure 1. The boxplots represent the distribution of data of all counties under the regions of the North East, North west, Yorkshire and The Humber, East and West Midlands, East England, London, South East, South West and Wales.

The resultant data after cleaning still appears to contain some outliers as they did not satisfy the condition to have a z-score of less than 3 in all columns. The boxplots for violence with injury shows that there is a skewness to the data. Most of the data will be found to the left of the median whereas the tail of the distribution will be found in the higher crime numbers skewed to the right. Though this doesn't necessarily suggest outliers, it does mean that most of the recorded crimes are lower and near the median and that areas of higher crime are less frequent. The boxplots for stalking and harassment also appear to have a skewness to the right, with the difference that the median is more centrally located between quartiles 1 and 3 which reflects that most of the data is equally partition either side of the median point. A possible observation is that there is seasonality with the crimes recorded. Studies have been conducted on checking seasonal oscillations with crime and anecdotal evidence suggests that there is a relationship. A study by (Linning, S. J., Andresen, M. A., & Brantingham, P. J. 2017) into property crime found that there are more spikes in crime with areas with larger variations in the weather and higher crime rates during the summer months. Furthermore, a study by (Tiihonen *et al.* 2017) found that there was a there was a relationship between ambient temperatures and higher violent crime rates in Finland. Thus, the two studies suggest that there is a likelihood that crimes are affected by seasonal stimuli.

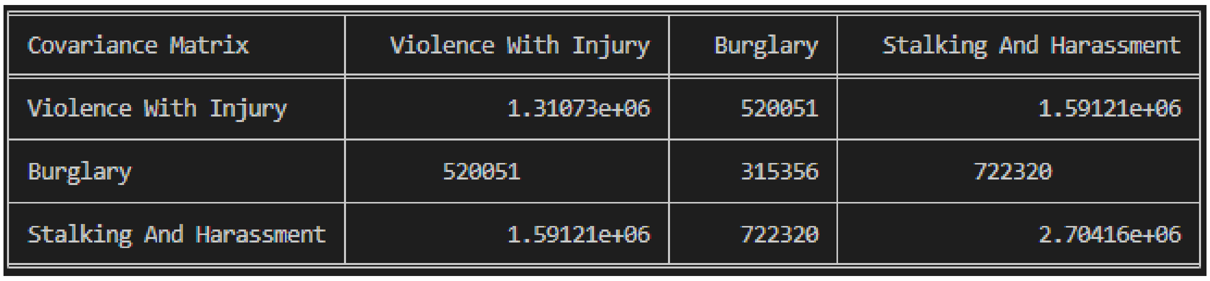
However, it can be said that the first 12 months of the data is lower with respect to one year on. A potential reason for this was the introduction of Covid-19 lockdown restrictions. The first lockdown occurred on the 23rd of March 2020 and the 24th of February saw the living with covid plan being published (Sherrington 2022). Thus, the initial 12 months of the data could be considered in lockdown, and then post-lockdown for the remainder. Within one week of the first lockdown, crime figures had significantly declined (Halford *et al.* 2020, Stickle & Felson 2020) marking an overall reduction in crime and is noticeable in the violence with injury plot in figure 1. For the first month after lockdown, crime decreased by 25% (National Police Chiefs' Council 2020). As lockdown restrictions were reduced, crime rates began to rise (Langton, Dixon & Farrell 2021), noticeable in the data for the January to March 2021 periods onwards.



crime	mean	median	std	var
Violence With Injury	2330.61	2054	1142.99	1.30643e+06
Burglary	1048.4	884	560.644	314322
Stalking And Harassment	3122.41	2843	1641.74	2.6953e+06

Figure 2. Statistical measures for crimes recorded between April 2020 - March 2022.

From these statistics, some observations can be made. It can be said that for a three-month period, a county has above or below average number of crimes if it is ± one standard deviation. So, for violence with injury crimes, a county is outside of the expected range if it has recorded more than 3474 crimes or less than 1188 crimes. The same applies to burglary where a county is receiving an irregular number of crimes if it is 1048.40 ± 560.64 and stalking and harassment if it receives 3122.41 ± 1641.74.



Covariance Matrix	Violence With Injury	Burglary	Stalking And Harassment
Violence With Injury	1.31073e+06	520051	1.59121e+06
Burglary	520051	315356	722320
Stalking And Harassment	1.59121e+06	722320	2.70416e+06

Figure 3. Covariance matrix of crimes against crimes in the dataset

The top-left to bottom-right diagonal of the data is not relevant in this case as it looks at the covariance against an identical category. However, it does show that for the other possible combinations, each crime tends to move in the same direction. For example, it may suggest that if burglary were to increase, then stalking and harassment would likely move in the same direction. Though this may not always be the case. Thus, correlation must be considered.

Hypothesis

Hypothesis question one

Did the Covid-19 lockdown restrictions effect the number of recorded violent crimes and burglaries?

H_0:

$$H_0: \mu_1 \approx \mu_2$$

$$H_0: \mu_3 \approx \mu_4$$

Assumption: Violence with injury means were similar, and burglary means were similar

H_1:

$$\mu_1 > \mu_2,\ \mu_3 < \mu_4$$

$$or$$

$$\mu_1 < \mu_2,\ \mu_3 > \mu_4$$

Assumption one: Violent crime is to have decreased during lockdown while burglary increased

Assumption: Violent crime is to have increased during lockdown while burglary decreased

H_2:

$$\mu_1 > \mu_2, \; \mu_3 > \mu_4$$
$$or$$
$$\mu_1 < \mu_2, \; \mu_3 < \mu_4$$

Assumption one: Violent crime and burglary increased

Assumption two: Violent crime and burglary decreased

H_3:

$$\mu_1 \approx \mu_2, \; \mu_3 < \mu_4$$
$$or$$
$$\mu_1 \approx \mu_2, \; \mu_3 > \mu_4$$

Assumption one: Violent crime is to have decreased during lockdown while burglary increased

Assumption two: Violent crime is to have increased during lockdown while burglary decreased

H_4:

$$\mu_1 > \mu_2, \; \mu_3 \approx \mu_4$$
$$or$$
$$\mu_1 < \mu_2, \; \mu_3 \approx \mu_4$$

Assumption one: Violent crime and burglary increased

Assumption two: Violent crime and burglary decreased

Where:

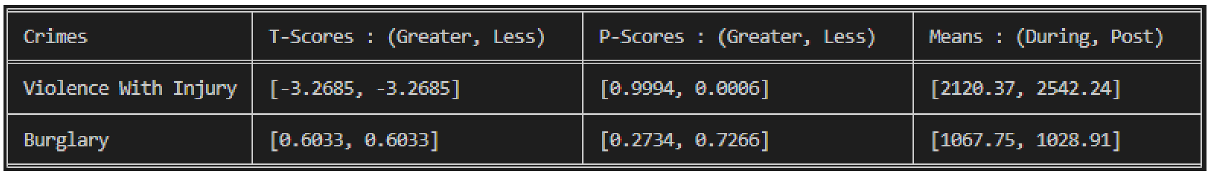
μ_1 = Mean violence with injury crime between April 2020 and March 2021

μ_2 = Mean violence with injury crime between April 2021 and March 2022

μ_3 = Mean burglary crime between April 2020 and March 2021

μ_4 = Mean burglary crime between April 2021 and March 2022

Testing hypothesis one

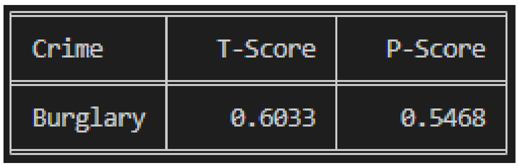


Crimes	T-Scores : (Greater, Less)	P-Scores : (Greater, Less)	Means : (During, Post)
Violence With Injury	[-3.2685, -3.2685]	[0.9994, 0.0006]	[2120.37, 2542.24]
Burglary	[0.6033, 0.6033]	[0.2734, 0.7266]	[1067.75, 1028.91]

Figure 4. T-test scores for time of April 2020 to March 2021 against April 2021 to March 2022. Greater or less denotes the one-sided test type.

The results were produced using a one-sided test, testing whether the mean of the first sample was greater or less than the second sample. The t-test null hypotheses changes for each type conducted. When testing if lockdown crime mean was less than post-lockdown the null hypothesis would be $u_{lockdown} > u_{post-lockdown}$ and alternate hypothesis would be $u_{lockdown} < u_{post-lockdown}$. If the p-value was not statistically significant, the null hypothesis cannot be rejected. However, the p-value for this was 0.0006 and is less than a 1% significant level (α = 0.01) which allows the null hypothesis to be rejected. This shows that there is very strong evidence that the mean crimes post-lockdown was higher than during lockdown for violence with injury crimes.

When testing burglary, the p-value was not within the accepted range to be able to reject a null hypothesis for both test cases even if the there was a 10% significance which is high. However, the mean does appear to be slightly less going from 1067.75 to 1028.91 in the post-lockdown period.



Crime	T-Score	P-Score
Burglary	0.6033	0.5468

Figure 5. T-score and p-score of a two-tailed test for burglary.

As the p-scores for one-tailed and two-tailed tests were more than 0.1, then there is no evidence to suggest that the means are significantly statistically different for burglary during lockdown and post-lockdown. The assumed null hypothesis of a two tailed test if the p-value is not statistically significant is that $H0: u_1 = u_2$ but as the mean is slightly different, then it could be said that $H0: u_1 \approx u_2$.

From the t-testing, the null hypothesis can be rejected as the violence with injury means were different and not roughly equal although the burglary crime means were almost equal. This satisfies the alternate condition of hypothesis four, where the mean of violence with injury was lower than that after restrictions were eased, and the burglary crime mean was roughly the same during and after lockdown restrictions.

Hypothesis question two

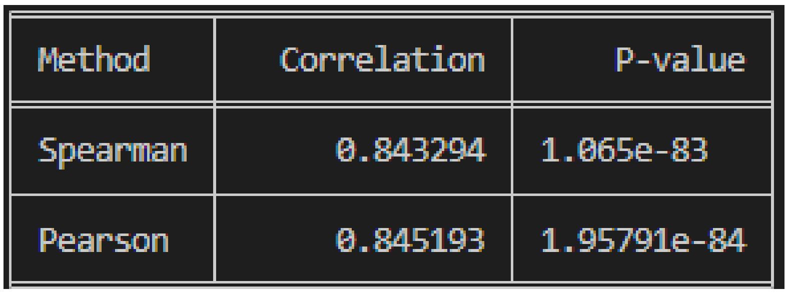
Is there a relationship between the amount of violence with injury crimes and stalking and harassment?

$$H_0: p = 0$$

$$H_1: p \neq 0$$

Where:

p is the correlation coefficient.



Method	Correlation	P-value
Spearman	0.843294	1.065e-83
Pearson	0.845193	1.95791e-84

Figure 6. Spearman and Pearson results for violence with injury against stalking and harassment.

Both Spearman and Pearson tests show a similar corelation with a marginal difference in the p-value. However, with a significance level of 1% (α = 0.01), the p-value is far below the threshold to reject the null hypothesis as there is sufficient evidence to claim a linear correlation as the correlation is significantly closer to one than zero. Thus, the alternate hypothesis can be accepted.

Model Implementations

Linear regression model

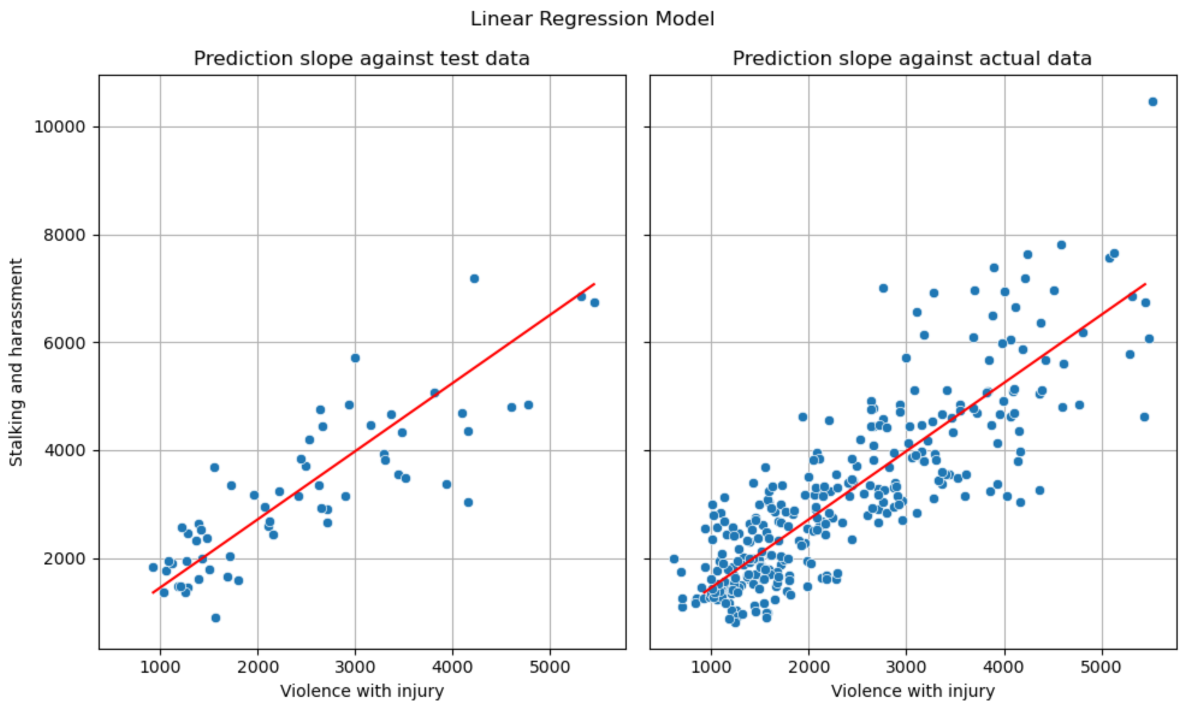


Figure 7. Linear regression model trained on violence with injury against stalking and harassment.

The linear regression model was trained on 80% of the data and then tested on the remainder. Although it had a high RMSE score of 808.2, it had a coefficient of determination score of 0.72 and can potentially be used to find missing information in data if only one of the used axis were available. Although a linear regression model was used, it would also be functionally possible to use another model such as a gradient boosting tree or random forest regression. However, a linear regression model is much faster to compute than the other regression models which is why it was used in this case. Due to the previously established correlation, accuracy was not that crucial as the values tend to deviate more from the line of best fit the larger the x and y value became.

Time-series forecasting model

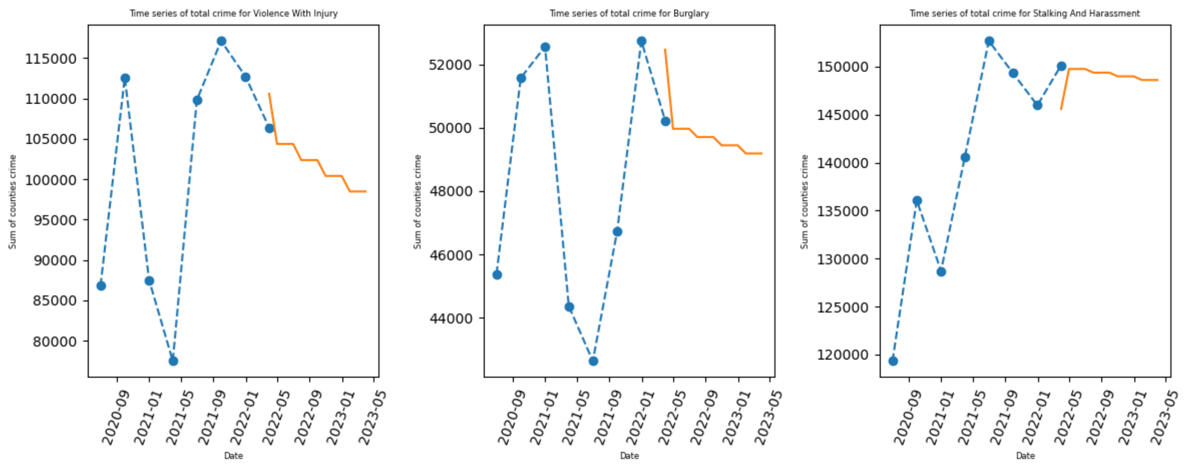
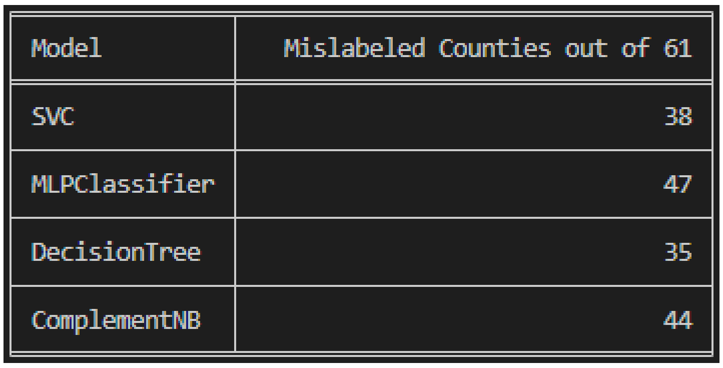


Figure 8. Sarimax model trained on the 8 data points available from the dataset.

As there were only 8 time points in the data, it was unlikely that a time series forecast would be accurate. Plotting a sarimax with quartile seasonal separation shows that the data it learned from was not sufficient as it does not appear to follow any trend that exists from the previous data. The sarimax model was trained across all data unlike the usual case where a portion of the data, usually around 20% is set aside to validate the model.

Classification model

A region column was applied to the data which was based on the county name. The new feature would introduce an aspect of categorical data that would allow for classification. Four classification models were implemented and tested to see which performed best. The features that each model learned from was all three of the crimes thus far. The goal was to be able to identify which electoral region a county may be from given the data for those three crimes.



Model	Mislabeled Counties out of 61
SVC	38
MLPClassifier	47
DecisionTree	35
ComplementNB	44

Figure 9. A chart showing the number of mislabelled counties for each model.

Model Comparisons

Due to the limited time and categorical data, it was difficult to approach it with machine learning methods. For a classification model, there needed to be some categorical value for it to be able to find discrete values based on a criterion of features. An attempt was made to create a categorical attribute based on which region the county was in, but the results showed that all models used performed poorly. The best model labelled 26 correctly out of 61 which is not high enough to be considered reliable. For time-series forecasting, it will provide more accurate predictions the more data it has to learn on. This can be said for most machine learning algorithms. However, the data only had 8 points in time to learn from which was added during the data preparation. The SARIMAX model was able to learn from the data but was not able to make meaningful predictions on past or future time points. The most efficient and accurate model appeared to be the linear regression as it focused on existing features in the data and could use the vast number of records to aid in the training. For this dataset, it can be said that regression models will be more effective on the types of features than classification or time series forecasting.

References

Halford, E., Dixon, A., Farrell, G. *et al*. (2020). Crime and coronavirus: social distancing, lockdown, and the mobility elasticity of crime. *Crime Science*. Vol 9, (11). https://doi.org/10.1186/s40163-020-00121-w (Accessed: 29/12/2022)

Langton, S., Dixon, A. & Farrell, G. (2021) Six months in: pandemic crime trends in England and Wales. Crime Science. Vol 10, (6). https://doi.org/10.1186/s40163-021-00142-z (Accessed: 02/01/2023)

Linning, S. J., Andresen, M. A., & Brantingham, P. J. (2017). Crime Seasonality: Examining the Temporal Fluctuations of Property Crime in Cities With Varying Climates. *International Journal of Offender Therapy and Comparative Criminology*. Vol 61, (16), pp.1866–1891. https://doi.org/10.1177/0306624X16632259 (Accessed: 31/12/2022)

National Police Chiefs' Council. (2020). Sustained falls in recorded crime reported throughout lockdown. Available at: https://news.npcc.police.uk/releases/sustained-falls-in-recorded-crime-reported-throughout-lockdown (Accessed: 04/01/2023)

Sherrington, A. (2022) 2 Years of COVID-19 on GOV.UK. *Government Digital Service*. Available at: https://gds.blog.gov.uk/2022/07/25/2-years-of-covid-19-on-gov-uk/ (Accessed: 02/01/2023)

Stickle, B., Felson, M. (2020) Crime Rates in a Pandemic: the Largest Criminological Experiment in History. *American Journal of Criminal Justice*. Vol 45, pp.525–536. https://doi.org/10.1007/s12103-020-09546-0 (Accessed: 03/01/2023)

Tiihonen, J., Halonen, P., Tiihonen, L. et al. (2017). The Association of Ambient Temperature and Violent Crime. *Scientific Reports*. Vol 7, (6543). https://doi.org/10.1038/s41598-017-06720-z (Accessed: 31/12/2022)

Appendices

```
# ================================================
# --- Data Manipulation ---
import os
import pandas as pd
import numpy as np
from datetime import datetime

# ================================================
# --- Plotting ---
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from tabulate import tabulate

# ================================================
# --- Models ---

from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import ComplementNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from pandas.tseries.offsets import DateOffset
from statsmodels.tsa.statespace.sarimax import SARIMAX
# ================================================
# --- Configs ---
pd.set_option('display.max_rows', None)
```

```python
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 1000)
plt.rc('axes', axisbelow=True)
file_path = os.path.dirname(__file__)
os.chdir(file_path)
data_file_path = "../Datasets/Excel"
years = os.listdir(data_file_path)
page_name = "Table P1a"
# ================================================

def read_xlsx(file_name : str, format_type : int):
    df = pd.read_excel(file_name, sheet_name=page_name, index_col=None,
header=None)

    if format_type == 0:
        # --- Rows and Columns to drop ---
        # Row numbers should be reduced by one to account for -1 offset in
python
        # Rows: 1,2,3,4,6,63,64,65,66,67,68,69,70,71,72,73
        # Columns: D (4), W (24)
        df = df.iloc[:61, :]
        row_drops = [0, 1, 2, 3, 5]
        col_drops = [3, 22]
        # Row 4, col 2 is 1 row too high so it will be dropped down 1 row
        # and then row 0 removed as the rest is NaN values.
        df.iat[4,2] = df.iat[3,2]
        df.drop(index=row_drops, axis=0, inplace=True)
        df.drop(columns=col_drops, axis=1, inplace=True)
        df.columns = df.iloc[0, :]
        df.reset_index(drop=True, inplace=True)
        df.drop(index=0, axis=0, inplace=True)

    elif format_type == 1:
        # --- Rows and Columns to drop ---
        # Newer format does not need any column drops
        # Headers are on row 7 so all rows of the dataframe up to 6 will be
dropped
        df = df.iloc[:62, :]
        row_drops = [0, 1, 2, 3, 4, 5]
        df.drop(index=row_drops, axis=0, inplace=True)
        df.reset_index(drop=True, inplace=True)
        df.columns = df.iloc[0, :]
        df.drop(index=0, axis=0, inplace=True)

    return df

def remove_num_from_string(input_string : str, remove_commas : bool):
    formatted_list = []
    capitalise_next = False
    delete_next = False
    for letter in input_string:
        if letter == '[':
            break

        if letter.isdigit() == False and remove_commas == False and
delete_next == False:
            if letter != ' ':
                if letter == '\\':
                    delete_next = True

                else:
                    if capitalise_next:
                        formatted_list.append(letter.upper())
                        capitalise_next = False
                        delete_next = False

                    else: formatted_list.append(letter)

            elif letter == ' ':
                capitalise_next = True

        elif letter.isdigit() == False and remove_commas == True and letter !=
',':
            formatted_list.append(letter)

    return ''.join(formatted_list)

def convert_file_to_csv_format(file_path : str, format_type : int):
    df = read_xlsx(file_path, format_type)
    # Drop the numbers from the header names
    for i in range(0, len(df.columns)):
        df.rename(columns={df.columns[i] :
remove_num_from_string(str(df.columns[i]), False)}, inplace=True)

    df.columns.name = None

    return df

def transform_data(dataframe_input : pd.DataFrame, filter : bool) ->
pd.DataFrame:

    df = dataframe_input
    if filter:
        df = dataframe_input.filter(items=features_to_use)


    for i in range(len(df[df.columns[0]])):
        df[df.columns[0]].iloc[i] = 
remove_num_from_string(df[df.columns[0]].iloc[i], remove_commas=False)

    return df

def get_data_dataframe_list(filter : bool) -> list:
    police_data_dataframe_list = []
    date_index = 0
    for year in years:
        for excel_file in os.listdir(data_file_path + "/" + year):
            # First Character of file name signifies the format
            format_type = int(excel_file[0])
            # Returns a df from csv format and adds the df location to a list 
for
            # accessing later on.
            temp_df = convert_file_to_csv_format(data_file_path + "/" + year + 
"/" + excel_file, format_type)
            temp_df.drop(columns='AreaCode', inplace=True)
            temp_df = transform_data(temp_df, filter)
            temp_df.set_index(temp_df.columns[0], inplace=True)
            for column in temp_df.columns:
                temp_df[column] = temp_df[column].astype(dtype='float')


            temp_df["Date"] = 
pd.to_datetime(datetime.strptime(date_range[date_index], "%d/%m/%Y"), 
format='%d%m%Y')
            temp_df["Date"] = temp_df['Date'].dt.date
            police_data_dataframe_list.append(temp_df)
            date_index += 1


    return police_data_dataframe_list

features_to_use = ['AreaName', 'ViolenceWithInjury', 'Burglary', 
'StalkingAndHarassment']

date_range = ['30/06/2020', '30/09/2020', '31/12/2020', '31/03/2021', 
    '30/06/2021', '30/09/2021', '31/12/2021', '31/03/2022']

date_range_labels = ['apr-jun-2020', 'jul-sep-2020', 'oct-dec-2020', 'jan-mar-
2021',
    'apr-jun-2021', 'jul-sep-2021', 'oct-dec-2021', 'jan-mar-2022']



def get_joined_data(frame_list : list) -> pd.DataFrame:
    joined_dfs = pd.DataFrame()
    for df in frame_list:
        joined_dfs = pd.concat([df, joined_dfs], ignore_index=False)

    return joined_dfs

def get_feature_stats(data_frame : pd.DataFrame) -> pd.DataFrame:
    mean = [np.mean(data_frame[feature]) for feature in features_to_use[1:]]
    median = [np.median(data_frame[feature]) for feature in
features_to_use[1:]]
    std = [np.std(data_frame[feature]) for feature in features_to_use[1:]]
    var = [np.var(data_frame[feature]) for feature in features_to_use[1:]]
    results = pd.DataFrame({'crime' : [split_capitals(feature) for feature in
features_to_use[1:]], 'mean' : mean,
        'median' : median, 'std' : std, 'var' : var})

    results.set_index('crime', drop=True, inplace=True)
    return results

def get_covariance_stats(data_frame : pd.DataFrame) -> pd.DataFrame:
    covar_df = pd.DataFrame(columns=[split_capitals(col) for col in
data_frame.columns], index=[split_capitals(col) for col in
data_frame.columns])
    covar_df.index.name = 'Covariance Matrix'
    for col in range(len(covar_df.columns)):
        for row in range(len(covar_df.index)):
            covar_df.iat[row, col] =
float(np.cov(data_frame[data_frame.columns[col]],
data_frame[data_frame.columns[row]])[0, 1])

    return covar_df

areas_to_drop = ["ENGLANDANDWALES", "ENGLAND", "London,CityOf", "WALES",
"MetropolitanPolice"]
regions = ["NorthEast", "NorthWest", "YorkshireAndTheHumber", "EastMidlands",
"WestMidlands",
    "East", "London", "SouthEast", "SouthWest"]

def split_capitals(string: str) -> str:
    new_string = []
    for i in range(0, len(string)):
        if i == 0:
            new_string.append(string[i])

        else:
            if i == len(string) - 1:
                new_string.append(string[i])
                return ''.join(new_string)
            elif string[i].islower() == True and string[i+1].isupper() ==
True:
                new_string.append(string[i])
                new_string.append(" ")
                new_string.append(string[i+1])

            elif string[i].islower() == True and string[i+1].isupper() ==
False:
                new_string.append(string[i])


def boxplot_feature_report(joined_df : pd.DataFrame):

    fig, axes = plt.subplots(1, 3, sharey=False)
    fig.set_figwidth(12)
    fig.set_figheight(8)
    joined_df = joined_df.sort_values(by=['Date'], ascending=True)
    col = 0
    for crime in features_to_use[1:]:
        axes[col].grid(axis='y')
        # First Row
        sns.boxplot(data=joined_df, x='Date', y=crime, orient='v',
dodge=False, ax=axes[col])
        left, right = axes[col].get_xlim()
        bounds = [left, right]
        sns.lineplot(x=bounds, y=np.mean(joined_df[crime]), color='black',
linestyle='dashed', lw=1, ax=axes[col])
        axes[col].set_xticks(ticks=np.arange(0, len(date_range_labels)))
        axes[col].set_xticklabels(labels=date_range_labels, rotation=45,
ha='right')
        axes[col].set_title(f"{split_capitals(crime)} Statistics", fontsize=9)
        axes[col].set_ylabel("Recorded Crimes")

        bottom, top = axes[col].get_ylim()
        axes[col].annotate(xy=(0.2, bottom + 75), text='Mean line:
{0}'.format(round(np.mean(joined_df[crime]), 2)), horizontalalignment='left',
            fontsize=9, weight='semibold')

        col += 1

    fig.suptitle("County boxplots with mean line")
    fig.tight_layout()
    #plt.show()


def remove_outliers(dataframe : pd.DataFrame) -> pd.DataFrame:
    return
dataframe.iloc[(np.abs(stats.zscore(dataframe[features_to_use[1:]])) <
3).all(axis=1)]
```

```
def t_test(data_frame : pd.DataFrame, columns : list):

    t_score = []
    p_score = []
    means = []

    conditions = ['greater', 'less']

    date_list_lockdown = [pd.to_datetime('2020-06-30').date(),
pd.to_datetime('2020-09-30').date(), pd.to_datetime('2020-12-31').date(),
pd.to_datetime('2021-03-31').date()]
    date_list_lifted = [pd.to_datetime('2021-06-30').date(),
pd.to_datetime('2021-09-30').date(), pd.to_datetime('2021-12-31').date(),
pd.to_datetime('2022-03-31').date()]
    for column in columns:
        t_score_per_condition = []
        p_score_per_condition = []
        data_lockdown =
data_frame[column].loc[data_frame['Date'].isin(date_list_lockdown)].tolist()
        #print(len(data_lockdown))
        data_lifted =
data_frame[column].loc[data_frame['Date'].isin(date_list_lifted)].tolist()
        #print(len(data_lifted))
        means.append([round(np.mean(data_lockdown), 2),
round(np.mean(data_lifted), 2)])
        #print(data_lifted)
        for measure in conditions:
            scores = stats.ttest_ind(data_lockdown, data_lifted,
alternative=measure)
            t_statistic, p_val = scores
            t_score_per_condition.append(round(t_statistic, 4))
            p_score_per_condition.append(round(p_val, 4))

        t_score.append(t_score_per_condition)
        p_score.append(p_score_per_condition)

    results = pd.DataFrame({'T-Scores : (Greater, Less)' : t_score, 'P-Scores
: (Greater, Less)' : p_score, 'Means : (During, Post)' : means},
index=[split_capitals(column) for column in columns])
    results.index.name = 'Crimes'
    return results

def two_tail_burglary(data_frame : pd.DataFrame) -> pd.DataFrame:
    t_score = []
    p_score = []
    date_list_lockdown = [pd.to_datetime('2020-06-30').date(),
pd.to_datetime('2020-09-30').date(), pd.to_datetime('2020-12-31').date(),
pd.to_datetime('2021-03-31').date()]
    date_list_lifted = [pd.to_datetime('2021-06-30').date(),
pd.to_datetime('2021-09-30').date(), pd.to_datetime('2021-12-31').date(),
pd.to_datetime('2022-03-31').date()]
    data_lockdown =
data_frame['Burglary'].loc[data_frame['Date'].isin(date_list_lockdown)].tolist
()
    data_lifted =
data_frame['Burglary'].loc[data_frame['Date'].isin(date_list_lifted)].tolist()
    scores = stats.ttest_ind(data_lockdown, data_lifted, alternative='two-
sided')
    t_statistic, p_val = scores
    t_score.append(round(t_statistic, 4))
    p_score.append(round(p_val, 4))
    results = pd.DataFrame({'T-Score' : t_score, 'P-Score' : p_score},
index=['Burglary'])
    results.index.name = 'Crime'
    return results

def corr_test(data_frame : pd.DataFrame, col_1 : str, col_2 : str) ->
pd.DataFrame:
    spear_cor, spear_pval = stats.spearmanr(a=data_frame[col_1],
b=data_frame[col_2])
    pear_cor, pear_pval = stats.pearsonr(data_frame[col_1], data_frame[col_2])

    result = pd.DataFrame({'Correlation':[spear_cor, pear_cor], 'P-value' :
[spear_pval, pear_pval]}, index=['Spearman', 'Pearson'])
    result.index.name = 'Method'
    return result

class lin_regresser:
    def __init__(self, data : pd.DataFrame, col_1 : str, col_2 : str) -> None:
        self.model = LinearRegression()
        self.x = np.array(data[col_1].tolist()).reshape(-1, 1)
        self.y = np.array(data[col_2].tolist()).reshape(-1, 1)
        self.x_train, self.x_test, self.y_train, self.y_test =
train_test_split(self.x, self.y, test_size=0.2, random_state=42)
        self.y_pred = []
        None

    def train(self):
        self.model.fit(self.x_train, self.y_train)
        self.y_pred = self.model.predict(self.x_test)
        None

    def get_preds(self, x_values : list) -> float:
        x_values = np.array(x_values).reshape(-1, 1)
        res = np.array(self.model.predict(x_values)).flatten()
        return res

    def show_train_result(self):
        print(f"Y intercept: {self.model.intercept_}")
        print(f"Coefficient of Determination: {self.model.score(self.x_train,
self.y_train)}")
        print(f"Mean Absolute Error: {metrics.mean_absolute_error(self.y_test,
self.y_pred)}")
        print(f"Mean Squared Error: {metrics.mean_squared_error(self.y_test,
self.y_pred)}")
        print(f"Root Mean Squared Error:
{np.sqrt(metrics.mean_squared_error(self.y_test, self.y_pred))}")
        None

    def show_training_plot(self):
        fig, axes = plt.subplots(1, 2, sharex=True, sharey=True)

        axes[0].grid(True)
        sns.scatterplot(x=self.x_test.flatten(), y=self.y_test.flatten(),
ax=axes[0])
        sns.lineplot(x=self.x_test.flatten(), y=self.y_pred.flatten(),
color='red', ax=axes[0])
        axes[0].set_title("Prediction slope against test data")
        axes[0].set_xlabel("Violence with injury")
        axes[0].set_ylabel("Stalking and harassment")

        axes[1].grid(True)
        sns.scatterplot(x=self.x.flatten(), y=self.y.flatten(), ax=axes[1])
        sns.lineplot(x=self.x_test.flatten(), y=self.y_pred.flatten(),
color='red', ax=axes[1])
        axes[1].set_title("Prediction slope against actual data")
        axes[1].set_xlabel("Violence with injury")
        axes[1].set_ylabel("Stalking and harassment")

        fig.set_figwidth(10)
        fig.set_figheight(6)
        fig.suptitle("Linear Regression Model")
        fig.tight_layout()


region_mapper = pd.DataFrame({
    "NorthEast" : ['Cleveland', 'Durham', 'Northumbria', '', '', ''],
    "NorthWest" : ['Cheshire', 'Cumbria', 'Greater', 'Manchester',
'Lancashire', 'Merseyside'],
    "YorkshireAndTheHumber" : ['Humberside', 'NorthYorkshire',
'SouthYorkshire', 'WestYorkshire', '', ''],
    "EastMidlands" : ['Derbyshire', 'Leicestershire', 'Lincolnshire',
'Northamptonshire', 'Nottinghamshire', ''],
    "WestMidlands" : ['Staffordshire', 'Warwickshire', 'WestMercia',
'WestMidlands', '', ''],
    "East" : ['Bedfordshire', 'Cambridgeshire', 'Essex', 'Hertfordshire',
'Norfolk', 'Suffolk'],
    "London" : ['London', '', '', '', '', ''],
    "SouthEast" : ['Hampshire', 'Kent', 'Surrey', 'Sussex', 'ThamesValley',
''],
    "SouthWest" : ['AvonAndSomerset', 'DevonAndCornwall', 'Dorset',
'Gloucestershire', 'Wiltshire', ''],
    "Wales" : ['Dyfed-Powys', 'Gwent', 'NorthWales', 'SouthWales', '', '']
    })

def get_region_label(county : str) -> str:
    for column in region_mapper.columns:
        for i in range(0,6):
            if county == region_mapper[column].iat[i]:
                return column

    return None

def apply_region_label(data_frame : pd.DataFrame) -> pd.DataFrame:
    data_frame['Region'] = [get_region_label(county) for county in
data_frame.index]
    return data_frame

class region_svc_classifier:
    def __init__(self, data : pd.DataFrame, target_col : str) -> None:
        self.model = SVC()
        self.features = data[[column for column in data.columns if column not
in [target_col, 'Date']]]
        self.target = data[target_col]
        self.x_train, self.x_test, self.y_train, self.y_test =
train_test_split(self.features, self.target, test_size=0.2, random_state=42)
        self.y_pred = []
        None

    def train(self):
        self.model.fit(self.x_train, self.y_train)
        self.y_pred = self.model.predict(self.x_test)
        return (self.y_test != self.y_pred).sum() # returns quantity of
mislabeled points

    def get_preds(self, x_values : list) -> float:
        x_values = np.array(x_values).reshape(-1, 1)
        res = np.array(self.model.predict(x_values)).flatten()
        return res

class region_mlp_classifier:
    def __init__(self, data : pd.DataFrame, target_col : str) -> None:
        self.model = MLPClassifier()
        self.features = data[[column for column in data.columns if column not
in [target_col, 'Date']]]
        self.target = data[target_col]
        self.x_train, self.x_test, self.y_train, self.y_test =
train_test_split(self.features, self.target, test_size=0.2, random_state=42)
        self.y_pred = []
        None

    def train(self):
        self.model.fit(self.x_train, self.y_train)
        self.y_pred = self.model.predict(self.x_test)
        return (self.y_test != self.y_pred).sum() # returns quantity of
mislabeled points

    def get_preds(self, x_values : list) -> float:

        x_values = np.array(x_values).reshape(-1, 1)
        res = np.array(self.model.predict(x_values)).flatten()
        return res

class region_tree_classifier:
    def __init__(self, data : pd.DataFrame, target_col : str) -> None:
        self.model = DecisionTreeClassifier(max_depth=30)
        self.features = data[[column for column in data.columns if column not
in [target_col, 'Date']]]
        self.target = data[target_col]
        self.x_train, self.x_test, self.y_train, self.y_test =
train_test_split(self.features, self.target, test_size=0.2, random_state=42)
        self.y_pred = []
        None

    def train(self):
        self.model.fit(self.x_train, self.y_train)
        self.y_pred = self.model.predict(self.x_test)
        return (self.y_test != self.y_pred).sum() # returns quantity of
mislabeled points

    def get_preds(self, x_values : list) -> float:

        x_values = np.array(x_values).reshape(-1, 1)
        res = np.array(self.model.predict(x_values)).flatten()
        return res

class region_comp_classifier:
    def __init__(self, data : pd.DataFrame, target_col : str) -> None:
        self.model = ComplementNB()
        self.features = data[[column for column in data.columns if column not
in [target_col, 'Date']]]
        self.target = data[target_col]
        self.x_train, self.x_test, self.y_train, self.y_test =
train_test_split(self.features, self.target, test_size=0.2, random_state=42)
        self.y_pred = []
        None

    def train(self):
        self.model.fit(self.x_train, self.y_train)
        self.y_pred = self.model.predict(self.x_test)
        return (self.y_test != self.y_pred).sum() # returns quantity of
mislabeled points

    def get_preds(self, x_values : list) -> float:

        x_values = np.array(x_values).reshape(-1, 1)
        res = np.array(self.model.predict(x_values)).flatten()
        return res

def plot_classifier_results(data : pd.DataFrame):
    svc_class = region_svc_classifier(data, 'Region')
    svc_errors = svc_class.train()
    mlp_class = region_mlp_classifier(data, 'Region')
    mlp_errors = mlp_class.train()
    tree_class = region_tree_classifier(data, 'Region')
    tree_errors = tree_class.train()
    comp_class = region_comp_classifier(data, 'Region')
    comp_errors = comp_class.train()

    results = pd.DataFrame({'Mislabeled Counties out of 61' : [svc_errors,
mlp_errors, tree_errors, comp_errors]}, index=['SVC', 'MLPClassifier',
'DecisionTree', 'ComplementNB'])
    results.index.name = 'Model'
    print(tabulate(results, headers='keys', tablefmt='fancy_grid'))

def time_series_plot(data : pd.DataFrame, columns : list):

    fig, ax = plt.subplots(1, 3)
    crime_sum = data.groupby('Date').sum().reset_index()
    crime_sum.set_index('Date', inplace=True, drop=True)

    ax_index = 0
    for crime in columns:

        sari_mod = SARIMAX(crime_sum[crime], seasonal_order=(0, 0, 0,
4),order=(1,0,0))
        sari_res = sari_mod.fit(disp=False, maxiter=250)
        future_time_periods = [crime_sum.index[-1] + DateOffset(months=x) for
x in range(0, 13, 1)]
        sari_predictions = np.array([sari_res.predict(time) for time in
future_time_periods]).flatten()
        ax[ax_index].plot(crime_sum[crime], marker = 'o', linestyle='dashed')
        sns.lineplot(x=future_time_periods, y=sari_predictions,
ax=ax[ax_index])
        ax[ax_index].set_xlabel("Date", fontsize = 6)
        ax[ax_index].set_ylabel("Sum of counties crime", fontsize = 6)
        ax[ax_index].set_title(f"Time series of total crime for
{split_capitals(crime)}", fontsize = 6)
        ax[ax_index].tick_params(labelrotation = 70, axis='x')
        ax_index+=1

    fig.set_figwidth(12)
    fig.tight_layout()
    #plt.show()

def main():

    data_list = get_data_dataframe_list(filter=True)
    df = get_joined_data(data_list)
    df = data_list[0]
    counties = [df.index[i] for i in range(0, len(df.index)) if df.index[i]
not in regions]

    for area in areas_to_drop:
        counties.remove(area)

    all_data = get_joined_data(data_list) # Join data from data list
    county_data = all_data.loc[counties]
    county_data = remove_outliers(county_data)
    boxplot_feature_report(county_data)

    time_series_plot(county_data, features_to_use[1:])

    results = corr_test(county_data, features_to_use[1], features_to_use[-1])
    print(tabulate(results, headers='keys', tablefmt='fancy_grid'))

    stat_df = get_feature_stats(county_data)

    print(tabulate(stat_df, headers='keys', tablefmt='fancy_grid'))
```

```python
    print(tabulate(get_covariance_stats(county_data[[col for col in
county_data.columns if col != 'Date']]), headers='keys',
tablefmt='fancy_grid'))

    results = t_test(data_frame=county_data, columns=['ViolenceWithInjury',
'Burglary'])
    print(tabulate(results, headers='keys', tablefmt='fancy_grid'))

    burglary_data = two_tail_burglary(county_data)
    print(tabulate(burglary_data, headers='keys', tablefmt='fancy_grid'))

    lin_reg = lin_regresser(county_data, col_1='ViolenceWithInjury',
col_2='StalkingAndHarassment')
    lin_reg.train()
    lin_reg.show_training_plot()

    county_data = apply_region_label(county_data)
    plot_classifier_results(county_data)

    plt.show()

if __name__ == "__main__":
    main()
```