Crime Classification Analysis

Hruthik Vinnakota

Jugal Kishore Ruvva

Rajeev Yenugula

Shilpa Bai Udaya Singh

Department of Data Analytics, San Jose State University

DATA 228 - Big Data Technologies and Applications

Professor: Dr Ming - Hwa Wang

May 17, 2023

ACKNOWLEDGMENT

It has been an enormous honor and pleasure to be admitted into the M.S. in Data Analytics program at SAN JOSE STATE UNIVERSITY, San Jose. Prof. Dr. Ming Hwa Wang's excellent effort and helpful advice helped us complete this research, and we are extremely appreciative. We also want to express our gratitude to everyone who has supported us in any way to ensure the smooth completion of this project.

Contents

LIST OF FIGURES AND TABLES

FIGURES

TABLES

ABSTRACT

Crime classification is an important task for law enforcement agencies as it helps in identifying patterns and trends in criminal activities. The use of crime classification analysis has led to significant improvements in the efficiency and effectiveness of criminal enforcement organizations. This process involves the use of advanced analytical techniques to identify patterns and relationships between different types of crimes and the factors that contribute to them. Machine learning algorithms have been widely used for this purpose, as they can analyze large volumes of data and identify patterns and relationships between different types of crimes. In this project, we propose to develop a machine learning-based system for crime classification that can create focused interventions and preventative tactics that are more likely to be helpful in preventing and solving crimes by creating accurate and reliable models that can anticipate the type and severity of a crime based on various factors such as the location, time, and modus operandi. We are intrigued by an approach that classification of crimes into multiple categories. Using crime analysis, we may examine many crime categories and determine whether the real-time data and incidents accurately match the crime description. We examine the relationship between crime category and incident code description by using machine learning concepts to crime data taken from official government databases. The conclusion of the paper demonstrates that there is a significant link between the incident category and crime characterization listed from the vast data based on the law enforcement registry.

Introduction

A crime description is automatically classified into a specific category based on its characteristics via crime classification. This system can be used by law enforcement agencies to select the appropriate officers for a particular crime or to automatically assign officers in accordance with the crime's classification. The system would accept a crime description as input and employ machine learning algorithms to evaluate the text in order to find specific keywords, terms and tendencies that can be used to categorize the crime. A sizable collection of crime descriptions and their accompanying categories would be used to train the algorithm. The machine learning algorithms would be taught how to recognize the traits of various crimes and accurately classify them into the appropriate categories using the dataset. This can help law enforcement operations run more smoothly and efficiently by saving time and resources. Eventually, lower crime rates in their neighborhoods.

Objective

The project will use big data techniques to analyze the vast amount of crime data that is collected by law enforcement agencies. This data will be preprocessed and used to train the machine learning algorithms to accurately classify the crime descriptions into different categories, such as assault, theft, burglary, etc. The developed system will be able to take in a new crime description and automatically classify it into a category based on the learned patterns and relationships in the data. This will not only save time for law enforcement agencies but also increase the accuracy of assigning officers to investigate the crime. Ultimately, enhancing public safety in the long run. In this process we are going to analyze the crime data will involve several stages, including data collection, preprocessing, feature extraction, model training, and testing. The system's performance will be evaluated using standard evaluation metrics, such as precision,

recall, and F1 score, to ensure that it is accurate and reliable. Then convey the results of the investigation using graphics.

Problem Statement

Due to a number of factors. Firstly, there may be a lower chance of a crime being solved as a result of the frequent delays in assigning enforce to investigate cases. Secondly, Traditional methods of analyzing this data are often slow and inefficient, leading to delays in the investigative process. Thirdly, the current process of assigning officers to investigate crimes based on crime descriptions can be error-prone, resulting in officers being assigned to investigate crimes outside of their expertise or experience. Furthermore, the lack of a standardized crime classification system across law enforcement agencies can result in inconsistencies in the way crimes are classified and investigated. This can lead to inefficiencies and inaccuracies in the investigative process.

Project Significance

After gathering the San Francisco Crimes dataset from Kaggle, we carried out additional searches on the websites of law enforcement. For this project, we'll be using a dataset, which is a CSV data file with information about the crime category and other classification-related information, such as the location and date. For the purpose of developing the prediction model, we will be obtaining these data from law enforcement agency websites for various crime categories. It is an effective choice for a big data analytics project because it is anticipated that the dataset will be quite huge overall. We will perform various tasks in this project, including data pre-processing, feature extraction, and model development, using PySpark API, as well as other libraries. PySpark is a Python library that provides an interface for Apache Spark, a distributed computing framework used for processing large-scale data sets. By leveraging

PySpark's powerful capabilities, we can efficiently analyze and process the large amount of crime data collected by law enforcement agencies.

Additionally, we will use other relevant libraries to supplement PySpark's capabilities, such as scikit-learn for machine learning algorithms, pandas for data manipulation, and matplotlib for visualization. These tools will enable us to effectively preprocess the data, extract relevant features, and develop accurate machine learning models for crime classification. As a result, this project is extremely pertinent to this course.

Prior Methods

There are a variety of approaches that can be used to classify crime descriptions into different categories, each with its own strengths and limitations. The choice of approach will depend on the specific requirements of the project, including the complexity of the crime descriptions, the availability of labeled data, and the resources available for computation and training. Initially, Rule-based systems require developing a series of rules or decision trees to categorize crimes in accordance with certain criteria, such as the scene of the crime or the kind of weapon used. While this strategy may work in some circumstances, it might not be adaptable enough to handle complex criminal descriptions or deal with new forms of crimes. Later, the significant information from crime descriptions can be extracted using NLP algorithms. For instance, NLP approaches to find the relationships between the entities (such persons, places, and things) stated in a crime report as well as the entities themselves. The offense can then be divided into many groups using this information. Even though this method can handle complicated criminal descriptions, it might need a lot of computational capacity and NLP technique experience.

Moreover, Since the information only contains citations that are connected to event reports, it is possible that it does not give a complete picture of policing and crime. For instance, unless it results in an arrest or similar occurrence that necessitates an incident report, a simple citation like a speeding ticket is typically not documented in the incident reports. In light of this, it is possible that not all types of policing and crime data are included in the incident reports dataset. Which is not a better approach. Our work is supported by keyword-based systems. To categorize crimes into distinct groups, we use keyword-based systems which use a list of terms or phrases. For instance, some phrases like "robbery" or "theft" might be connected to the "property crime" category, making this method useful and fruitful for determining the classification of a certain crime. Especially, when the crime descriptions are relatively simple, and the keywords can accurately capture the relevant information.

Why our proposal is a superior strategy?

According to our dataset, Incident reports may be excluded from the dataset due to court orders to seal records or for administrative reasons such as ongoing internal affairs investigations or criminal investigations. The data provided does not contain any personally identifiable information about individuals involved in the dataset, including suspects, victims, reporting parties, officers, witnesses, and others. Over and above, the purpose of this paper is to provide a comparative analysis of various models for predicting the crime category and criminal description and to present a comprehensive report on why one model would be superior than another, which was not included in any prior papers.

Area or scope of investigation

The project will aim to develop a user-friendly interface for the crime classification system that can be used by law enforcement agencies to input crime descriptions and receive

accurate crime category classifications. Our responsibility is to develop and train a machine learning model that can accurately classify crime descriptions into 39 pre-defined categories. We will then evaluate the accuracy of the model and deploy it into a production environment. The goal is to develop a system that can automatically assign a new crime description to one of the 39 categories.

The training process will involve preparing a dataset of crime descriptions and corresponding categories, preprocessing the data to remove irrelevant information and transform the text data into a suitable format for machine learning algorithms, and training the model on this preprocessed data. Once the model has been trained, we will test its accuracy using a separate dataset of crime descriptions and categories that the model has not been trained on. We will evaluate the model's performance using metrics such as precision, recall, and F1 score. We will also use cross-validation techniques such as k-fold cross-validation to evaluate the performance of the different machine learning algorithms and choose the best one for the task at hand. Additionally, we may perform hyperparameter tuning on the selected algorithm to optimize its performance. At the end, we will deploy the trained model into a production environment, where it will be used to classify new crime descriptions automatically. The system will take in new crime descriptions as input and assign them to one of the 39 pre-defined categories, allowing law enforcement agencies to allocate resources more effectively and handle cases more efficiently.

Theoretical Bases and Literature Review

Definition of the Problem

Criminal offenses are categorized based on their nature, severity, and other characteristics. This process is known as crime classification. In order to effectively allocate

resources, develop targeted strategies to prevent and control crime, and understand crime trends and patterns, law enforcement agencies, policymakers, and researchers use crime classification. Law enforcement organizations can prioritize their efforts and resources more effectively and respond to incidents by categorizing crimes differently.

Since no two crimes are alike and the criminal justice system must respond differently to various crimes, it is necessary to classify crimes. In order to create effective crime prevention and control strategies, it is crucial to be able to recognize crime patterns and trends. Additionally, it aids in identifying the types of crimes that are most common in a specific location, enabling law enforcement organizations to allocate resources appropriately. For instance, a different strategy is needed for property crimes like theft or burglary than for violent crimes like murder or assault. Law enforcement organizations can more effectively prioritize their efforts and resources and create focused strategies to prevent and look into various types of crimes by classifying crimes into different categories. Crime classification also offers a common language for discussing and reporting crimes, making it easier for law enforcement agencies to communicate and work together.

With big data techniques and machine learning, crime classification's effectiveness and precision can increase. Big data is the term used to describe the enormous volume of data produced from numerous sources, including social media, security cameras, and criminal databases. Machine learning algorithms can analyze these data to spot trends and patterns and automatically categorize crimes. Law enforcement organizations can quickly identify crime patterns and trends by analyzing massive amounts of data in real-time using machine learning algorithms. They can make better resource allocation decisions and increase public safety by using this information to develop focused strategies to prevent and investigate crimes. Machine

learning algorithms can also forecast future crime trends by learning from historical crime data. This can assist law enforcement organizations in actively preventing crimes and effectively responding to incidents.

Theoretical Background of the Problem

Criminology and data science are the theoretical foundations for crime classification using big data and machine learning. The study of crime, criminal behavior, and the criminal justice system is known as criminology. On the other hand, data science studies approach drawing conclusions and knowledge from data. When these two fields are combined, it can gain insights into crime trends and patterns that can be used to create strategies for reducing and preventing crime. The classification of crimes using big data analysis relies heavily on machine learning algorithms. In order to find patterns and trends in crime data and forecast future criminal activity, these algorithms use statistical models. This method is predicated on the idea that underlying patterns in the data can be found and applied to make predictions.

Using big data and machine learning, it is possible to classify crimes using two theoretical frameworks: the routine activity theory and the social disorganization theory. Routine activity theory is one theoretical framework that supports the classification of crimes using big data and machine learning. According to this theory, crime happens when three conditions are met: a motivated offender, an appropriate target, and the absence of a capable guardian. Law enforcement agencies can create strategies to prevent crime by examining crime data and spotting trends in these three areas.

The social disorganization theory is another theoretical framework for categorizing crimes using big data and machine learning. According to this theory, areas with high levels of social disorganization, such as poverty, unemployment, and social inequality, are more likely to

experience crime. Law enforcement agencies can create plans to reduce crime in these areas by analyzing crime data and spotting trends in these social factors.

Related Research

Related research on crime classification projects has explored using machine learning algorithms and big data analytics to identify factors contributing to crime rates. For example, Wang et al. (2016) in the paper "Crime Rate Inference with Big Data," the authors examine the application of big data analytics to crime rate inference. Based on publicly accessible data sources like Twitter and Google Street View, the authors suggest a methodology for calculating crime rates in various areas. The need for more standardization and the potential for biased data are two difficulties with using big data to infer crime rates covered in the paper. The authors compare their methodology to existing crime prediction models using data from several US cities and discover that it performs better. The paper thoroughly analyzes related research in big data-based crime rate prediction. The authors go over several methods employed to predict crime, including statistical modeling, machine learning, and social media analysis. The paper highlights the potential advantages of using big data for crime rate inference while outlining current methods' drawbacks. According to the author's methodology, policymakers and law enforcement organizations can better allocate resources for crime prevention and intervention with less waste.

The paper "Using Machine Learning for Prediction of Factors Affecting Crimes in Saudi Arabia" by Anadil Alsaqabi, Fatimah Aldhubayi, and Saleh Albahli aims to create a predictive model using machine learning algorithms to identify the factors that affect Saudi Arabia's crime rates. The Saudi Arabian General Authority for Statistics provided the authors with information on crime rates and socioeconomic factors like population density, unemployment, and poverty. The factors that influence crime rates were then predicted using a variety of machine learning

algorithms, such as logistic regression, decision trees, and support vector machines. The authors' thorough literature review on machine learning algorithms for crime classification offers a thorough overview of the field. The authors point out that less attention has been paid to identifying the factors that influence crime rates in previous research, which has mainly focused on using machine learning algorithms for crime prediction and hotspot identification. Additionally, the authors review the machine-learning techniques applied in earlier studies, such as decision trees, random forests, and artificial neural networks. The authors conclude by emphasizing the significance of choosing the right features and avoiding overfitting when creating successful predictive models using machine learning algorithms.

Supporting this research, the paper "Big Data-based Smart City Platform: Real-Time Crime Analysis" by Debopriya Ghosh, Soon Ae Chun, Nabil R. Adam, and Basit Shafiq focuses on the development of a big data-based smart city platform for real-time crime analysis. The authors highlight the need for an integrated platform to process and analyze large volumes of data from various sources as they discuss the opportunities and challenges of using big data analytics for crime analysis. The authors describe the platform's architecture and its various elements, such as data ingestion, storage, processing, and visualization. The authors' thorough review of the literature on big data analytics for crime analysis offers a thorough overview of the field. The authors review various data sources, such as social media, CCTV, and sensor networks, that can be used for crime analysis. They also highlight the various analytics methods, such as machine learning, data mining, and visualization, that can be applied to the study of crime. The authors also cover issues like data quality, privacy, and scalability that can arise when using big data analytics for crime analysis.

In contrast to other papers, "Big Crime Data Analytics and Visualization" by Mokhtar Mansour Salah and Xia Kewen provide a comprehensive survey of big data analytics and visualization techniques in crime classification. The paper also discusses various analytics methods like clustering, classification, and anomaly detection applied to crime prevention and prediction. The authors also draw attention to the difficulties in using big data for crime analysis, including issues with data privacy and the requirement for specialized knowledge and equipment. The paper thoroughly analyzes various visualization methods applied to crime analysis and investigation. According to the authors, the ability to spot patterns and relationships in large, complex data sets is one of the advantages of using visual analytics for criminal investigations. The paper discusses various visualization methods and tools applied to crime analysis, including network analysis, social media analysis, and geographic information systems.

Advantages and Disadvantages of Previous Research

The journal papers present novel methodologies for predicting and preventing crimes using big data analytics, machine learning techniques, and a thorough literature review. The papers discuss various data sources, analytics methods, and visualization tools that can be used for crime analysis, which can aid in improving the understanding of crime patterns and resource allocation for prevention and intervention among law enforcement agencies and policymakers. They offer insights into various crime classification topics, including forecasting crime rates, identifying the variables influencing crime, and examining crime trends in smart cities. They also suggest innovative frameworks and methods that can be used in various contexts to increase the efficiency and accuracy of crime classification. The papers also highlight the difficulties of using big data for crime analysis, including issues with data privacy and the requirement for specialized knowledge and equipment. The authors offer suggestions for overcoming these

difficulties, including using anonymized data and funding for law enforcement agency training programs that can be used in this project.

The scope and applicability of these papers are constrained, though, which is one of their drawbacks. One more potential drawback is that all the papers primarily concentrate on the technical aspects of crime prediction and prevention and may need to consider the social and ethical ramifications of using big data for crime analysis. When creating crime prediction models, these privacy and civil liberties concerns should be carefully considered, as surveillance cameras and social media data raise them. The papers might also not consider the possible biases that can be found in big data, such as racial and gender biases, which can result in incorrect predictions and maintain existing inequalities in the criminal justice system.

In addition, because the papers primarily concentrate on crime prevention and prediction in urban areas, they might not be applicable in rural areas where data sources and crime patterns may differ, and some of the papers may not be generalizable to other types of crime because they only address specific types of crime, like property crime or cybercrime.

Furthermore, some of the paper's methodology or data quality may have flaws. For instance, the study by Hongjian Wang et al. relies on data from crimes that were reported to the police, which may not include all criminal activity, especially that which is not reported to the authorities. Similarly, the paper by Mokhtar Mansour Salah et al. focuses on data visualization techniques, which might not be as successful as machine learning algorithms in identifying complex crime patterns. Last, the papers might need to fully consider the drawbacks of using machine learning and other analytics techniques, such as the need for human interpretation and expertise, which can reduce their usefulness in real-world scenarios.

Proposed Solution

Data Collection and Preparation.

The first step would be gathering and preparing the data for analysis. Information can be found in various places, including news articles, social media, crime reports, and surveillance cameras. Data collected from the San Francisco police department in geojson format with 2.14M rows and 14 columns will be used in this project. Preprocessing would be necessary to standardize the format, eliminate duplicate information, and fix errors. Demographic, economic, and social data may also be combined to find potential factors influencing crime rates.

Feature Selection and Model Building.

The second step would involve developing a machine learning model and choosing the appropriate features for categorizing crimes. The most pertinent features for categorizing crimes can be found using feature selection techniques like correlation analysis and principal component analysis. The features that help classification will be chosen carefully. The classification model can be constructed using a variety of machine learning algorithms, including decision trees, classification techniques, support vector machines, and neural networks.

Model Evaluation and Deployment.

The model's performance would be assessed before it was used to categorize crimes in the real world. Metrics such as accuracy, precision, recall, and F1 score can be used to assess the model. The model can be implemented in law enforcement organizations and integrated with current systems for real-time crime classification. The model can also be regularly updated with new data to enhance its efficacy and accuracy.

Model Differences

It differs from other solutions by utilizing the pyspark framework for distributed computing and built-in machine-learning libraries. To expand on that, the input file is a geojson file, which gives the scope of working with the latitude and longitude of the location of the crime. The data processing and feature selection will not be based on time features. Instead, it will be on the type of crime. Using pyspark allows access to additional feature extraction and selection techniques optimized for big data. The data visualization will be done only by using Python functions. Classification techniques will be used and compared to the regression techniques that are preached in reference journal papers. This project primarily focuses on understanding the data, processing the data, preparing the data, and crime classification.

Advantages of our Solution

The solution this project proposes uses Pyspark which is built for big data processing and can scale to handle large datasets. As a result, the proposed model can handle more crime data, which could increase the precision of the predictions and classifications.

Compared to conventional methods, Pyspark can process data in parallel, drastically cutting processing time. This implies that the model can process large amounts of data more quickly, offering analysis and response times that are almost real-time. Proposed machine learning models can guarantee the model's accuracy, effectiveness, and scalability.

Hypothesis

Hypothesis usually depends upon several factors of the model such as accuracy, precision, recall and f1. So, we need to come up with the model that gives high accuracy in classifying text to corresponding crime category. We have various input columns such as location, date, and time of occurrence of crime, category, and crime description. It is assumed

that data is balanced and of good quality and, such that there are less outliers, which otherwise could impact our classification. Also, it is of belief that data is equally distributed among all the columns and features are fairly correlated among each other which will be relevant for our classification. There is chance that, crime occurrences depend strongly on the some of the factors such as date and time, for example it is assumed that more crimes occur at month of the year or specific time of the day say midnight. Because of these dependencies we expect to find trend and relation among the various input features, which will help to get good accuracy. Along with that we hypnotize that if we consider input columns after performing careful feature engineering, when used in combination as features produces useful and meaningful results. For our project we assume that if we provide raw unstructured data that contain detailed description of crime incidents that has occurred at specific location and date and time, we could possibly extract output features containing the category of crime. Thus, we build a model such that it produces high accurate results.

Finally, we test our assumption using some of the suitable machine learning models such as Logistic regression, Naïve Bayes and SVM to train and classify new data. We assume that these models would not be impacted by change in data set size and provide better classification and results all the time. Finally, we will compare their performance based on evaluation metrics and baseline models.

Methodology

Collect input data.

Initially, we obtained the San Francisco Crimes dataset from Kaggle and then conducted further searches on police departments and government websites. Eventually, we were able to locate a reliable government website where we acquired the dataset in CSV and JSON format.

CSV (Comma-Separated Values) is a file format that stores and exchanges tabular data. In a CSV file, each row represents a record, and each column represents a feature. The values of the fields are separated by commas, hence the name Comma-Separated Values. The first row of a CSV file usually contains the headers or column names or features, while subsequent rows contain the data for each record. The dataset consists of incidents extracted from the SFPD Crime Incident Reporting system, covering a period ranging from January 1, 2003, to May 13, 2021. There are 9 features:

- Dates - timestamp of the crime incident
- Category - category of the crime incident (only in train.csv).
- Descript - detailed description of the crime incident (only in train.csv)
- DayOfWeek - the day of the week
- PdDistrict – the name of the Police Department District
- Resolution - how the crime incident was resolved (only in train.csv)
- Address - the approximate street address of the crime incident
- X - Longitude
- Y – Latitude

JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write, and for machines to parse and generate. JSON files are commonly used for data storage and exchange between interfaces. In a JSON file, data is represented as a collection of key-value pairs, where the keys are strings and the values can be strings, numbers, boolean, arrays, or objects. In Figure 1, the CSV file is uploaded i.e., Resilient Distributed Dataset (RDD) can be loaded using various methods depending on the dataset type like CSV or JSON. First will load the CSV file into a DataFrame and then convert it into an RDD.

Figure 1

Sample Dataset of Crime Classification

	A	B	C	D	E	F	G	H	I
1	Dates	Category	Descript	DayOfWeek	PdDistrict	Resolution	Address	X	Y
2	5/13/15 23:53	WARRANTS	WARRANT ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.42589	37.7745986
3	5/13/15 23:53	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.42589	37.7745986
4	5/13/15 23:33	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	VANNESS AV / GREENWICH ST	-122.42436	37.8004143
5	5/13/15 23:30	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	1500 Block of LOMBARD ST	-122.427	37.8008726
6	5/13/15 23:30	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	PARK	NONE	100 Block of BRODERICK ST	-122.43874	37.7715412
7	5/13/15 23:30	LARCENY/THEFT	GRAND THEFT FROM UNLOCKED AUTO	Wednesday	INGLESIDE	NONE	0 Block of TEDDY AV	-122.40325	37.7134307
8	5/13/15 23:30	VEHICLE THEFT	STOLEN AUTOMOBILE	Wednesday	INGLESIDE	NONE	AVALON AV / PERU AV	-122.42333	37.725138
9	5/13/15 23:30	VEHICLE THEFT	STOLEN AUTOMOBILE	Wednesday	BAYVIEW	NONE	KIRKWOOD AV / DONAHUE ST	-122.37127	37.7275641
10	5/13/15 23:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	RICHMOND	NONE	600 Block of 47TH AV	-122.50819	37.7766013
11	5/13/15 23:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	CENTRAL	NONE	JEFFERSON ST / LEAVENWORTH	-122.41909	37.8078016
12	5/13/15 22:58	LARCENY/THEFT	PETTY THEFT FROM LOCKED AUTO	Wednesday	CENTRAL	NONE	JEFFERSON ST / LEAVENWORTH	-122.41909	37.8078016
13	5/13/15 22:30	OTHER OFFENSES	MISCELLANEOUS INVESTIGATION	Wednesday	TARAVAL	NONE	0 Block of ESCOLTA WY	-122.48798	37.7376667
14	5/13/15 22:30	VANDALISM	MALICIOUS MISCHIEF, VANDALISM OF VEHICLES	Wednesday	TENDERLOIN	NONE	TURK ST / JONES ST	-122.41241	37.7830038
15	5/13/15 22:06	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	FILLMORE ST / GEARY BL	-122.43291	37.7843533
16	5/13/15 22:00	NON-CRIMINAL	FOUND PROPERTY	Wednesday	BAYVIEW	NONE	200 Block of WILLIAMS AV	-122.39774	37.7299347
17	5/13/15 22:00	NON-CRIMINAL	FOUND PROPERTY	Wednesday	BAYVIEW	NONE	0 Block of MENDELL ST	-122.38369	37.743189
18	5/13/15 22:00	ROBBERY	ROBBERY, ARMED WITH A KNIFE	Wednesday	TENDERLOIN	NONE	EDDY ST / JONES ST	-122.4126	37.783932
19	5/13/15 21:55	ASSAULT	AGGRAVATED ASSAULT WITH BODILY FORCE	Wednesday	INGLESIDE	NONE	GODEUS ST / MISSION ST	-122.42168	37.7428222
20	5/13/15 21:40	OTHER OFFENSES	TRAFFIC VIOLATION	Wednesday	BAYVIEW	ARREST, BOOKED	MENDELL ST / HUDSON AV	-122.3864	37.7389835
21	5/13/15 21:30	NON-CRIMINAL	FOUND PROPERTY	Wednesday	TENDERLOIN	NONE	100 Block of JONES ST	-122.41225	37.7825563
22	5/13/15 21:30	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	INGLESIDE	NONE	200 Block of EVELYN WY	-122.44939	37.7426688
23	5/13/15 21:17	ROBBERY	ROBBERY, BODILY FORCE	Wednesday	INGLESIDE	NONE	1600 Block of VALENCIA ST	-122.42027	37.7473316
24	5/13/15 21:11	WARRANTS	WARRANT ARREST	Wednesday	TENDERLOIN	NONE	100 Block of JONES ST	-122.41225	37.7825563
25	5/13/15 21:11	NON-CRIMINAL	STAY AWAY OR COURT ORDER, NON-DV RELATED	Wednesday	TENDERLOIN	NONE	100 Block of JONES ST	-122.41225	37.7825563
26	5/13/15 21:10	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	FILLMORE ST / LOMBARD ST	-122.43605	37.7998412
27	5/13/15 21:00	NON-CRIMINAL	LOST PROPERTY	Wednesday	TENDERLOIN	NONE	300 Block of OFARRELL ST	-122.41051	37.7860432
28	5/13/15 21:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	2000 Block of BUSH ST	-122.43102	37.7873881
29	5/13/15 21:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	INGLESIDE	NONE	500 Block of COLLEGE AV	-122.42366	37.7325565
30	5/13/15 21:00	LARCENY/THEFT	ATTEMPTED THEFT FROM LOCKED VEHICLE	Wednesday	TARAVAL	NONE	19TH AV / SANTIAGO ST	-122.47577	37.7449191
31	5/13/15 20:56	OTHER OFFENSES	MISCELLANEOUS INVESTIGATION	Wednesday	TARAVAL	NONE	2000 Block of 41ST AV	-122.49979	37.7485176
32	5/13/15 20:54	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	1300 Block of WEBSTER ST	-122.43105	37.7830296
33	5/13/15 20:50	NON-CRIMINAL	CIVIL SIDEWALKS, CITATION	Wednesday	MISSION	ARREST, BOOKED	400 Block of CASTRO ST	-122.43515	37.7617597
34	5/13/15 20:45	VANDALISM	MALICIOUS MISCHIEF, VANDALISM	Wednesday	NORTHERN	NONE	1500 Block of FILLMORE ST	-122.43274	37.7838425

Solving the Problem

Algorithm Design.

Crime classification is an important task for law enforcement agencies as it helps in identifying patterns and trends in criminal activities. With the increase in crime rates and the use of technology, it is becoming difficult to classify crimes manually. Therefore, there is a need to develop an automated system for crime classification. Objectives to develop Crime Classification are:

- To develop a machine learning model for crime classification that can accurately classify crimes based on their attributes.
- To reduce the time and effort required for manual crime classification by law enforcement agencies.
- To identify patterns and trends in criminal activities using classified data.

- To provide insights to law enforcement agencies to take preventive measures and reduce crime rates.

After identifying the problem statement, the next immediate step is to explore the data. Data exploration is an important step in the data analysis process as it helps in understanding the data and identifying patterns and trends that can inform the development of a crime classification model. It involves cleaning and preparing it for analysis, visualizing the data using charts and graphs, performing statistical analysis to identify correlations and relationships between variables, and creating new features from the existing data. Through data exploration, planning to gain insights into the data that can be used to develop an accurate and effective machine-learning model.

The next part is Data cleaning and preprocessing. crime classification as they ensure that the data is accurate, consistent, and ready for analysis. In this phase, the raw data is cleaned and transformed into a format that can be used for machine learning models. This involves handling missing or incorrect data, removing duplicates, and converting data types as necessary. It may also involve feature scaling or normalization to ensure that all features are on the same scale. Additionally, feature selection and engineering may be performed to extract the most important features for the classification model. The goal of data cleaning and preprocessing is to prepare the data for analysis so that the machine learning model can accurately classify crimes based on their attributes.

Feature engineering is an important step in crime classification as it involves creating new features from the existing data that can improve the accuracy and performance of the machine learning model. In the context of crime classification, some potential features that can be engineered from the crime description include the presence of specific keywords or phrases

that are indicative of certain types of crimes. For example, the presence of words like "theft", "robbery", or "assault" in the crime description could be used to classify the crime as a property crime, a violent crime, or a robbery.

As some of the longitude and latitude values in the dataset are empty, we intend to make use of the Google Maps API to fill in the missing values. The Google Maps API provides geocoding services that allow us to obtain the geographic coordinates of a location based on its address. By leveraging these geocoding services, we can retrieve the missing longitude and latitude values and complete the dataset. This can help us to improve the accuracy of the crime data and produce more reliable crime classification models. Utilizing the Google Maps API for filling in missing values is a common practice in data preprocessing and can lead to more robust and accurate analysis.

Each row of the dataset contains a Description of a specific crime, such as the crime classification, day of the week, police district, resolution, address, and coordinates of the location.

The goal is to develop an algorithm that can take a crime description as input and output the corresponding category of the crime. The crime categories in the dataset include Warrants, Other Offenses, Larceny/Theft, Vehicle Theft, and more.

To achieve this goal, the dataset can be preprocessed by cleaning and formatting the data. The crime descriptions can be tokenized and converted into a numerical representation using techniques such as bag-of-words or TF-IDF. A machine learning model, such as a classification algorithm like Decision Tree, Naive Bayes, SVM, and Random Forest are going to use and it can be trained on the preprocessed data to predict the crime category based on the crime description. Each algorithm has its strengths and weaknesses and is suited for specific types of crime

classification problems. The choice of the algorithm depends on the nature of the data and the problem at hand.

Language used.

In this Project, we will be using Pyspark as a programming language for data processing and analysis. Pyspark allows us to write code in Python and leverage the power of Spark's distributed computing engine to process large datasets in a scalable manner. Its scalability, ease of use, and wide range of data source support make it an attractive choice for organizations looking to extract insights from large datasets. Pyspark efficiency in handling large-scale data processing tasks is a significant advantage that makes it ideal for analyzing crimes in this project, given the large number of rows in the dataset.

Tools used.

The project uses two main tools - Jupyter Notebook, Google Cloud and Databricks. Jupyter Notebook is a web-based interactive computing platform that allows users to create and share documents with live code, text, visualizations, graphs, and plots. The platform provides a secure environment for sensitive data, as such data are not stored in local machines. Databricks, on the other hand, is a cloud-based data engineering tool that processes and transforms large amounts of data, making it easier to explore the data through machine learning models. This tool supports both structured and unstructured data, and it reduces batch processing time. Databricks employs data science to facilitate decision-making.

Generating Output

To validate the performance of predictive models built using machine learning techniques, we must use test data. The dataset is divided into train and test data, where the training is performed on the train data, and the validation is done on the test data to assess the

accuracy of the models. Performance metrics such as accuracy and precision are used to compare the performance of the models.

To make a classification, we check whether the prediction is favorable compared to the last non-predicted value and analyze the global polarity. Based on this analysis, appropriate classifications are made.

Testing Against Hypotheses

The project employs four prediction models, namely Decision Tree, Naive Bayes, SVM, and Random Forest which are evaluated using the same dataset to compare their performance. Finally, to compare the accuracy of different algorithms, one can use performance metrics such as precision, recall, F1-score, and confusion matrix. These metrics provide a quantitative measure of the algorithm's performance, allowing for a fair and objective comparison.

Code and Implementation:

Download python and spark latest version and install necessary packages including pyspark and related libraries. The source file for this project can be found at **https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-Historical-2003/tmnf-yvry.** Export the data as csv file.

Import these files into pyspark data frame as shown in the code (refer appendix for complete code) or import into Databricks for visualization.

Figure 2

Importing all necessary libraries in pyspark

```
import pyspark
spark = pyspark.sql.SparkSession.builder.appName("DATA228_Project").getOrCreate()

sc = spark.sparkContext
```

```
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer,T
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.classification import LogisticRegression, NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.mllib.evaluation import MulticlassMetrics
```

The code reads a CSV file SF_data.csv using Pyspark DataFrame API and sets various options for parsing the file. Specifically, it sets header to true to use the first row as the header, infer Schema to true to infer the schema of the DataFrame from the data, and timestamp to true to automatically parse any timestamp columns. The resulting DataFrame is stored in the variable df. The code df1=df.select(str.strip(PdId),Incident Code, Category, Descript, DayOfWeek, Date, Time, PdDistrict, X, Y) is selecting columns from the df dataframe using the select method. The str.strip() method is being applied to the string PdId, but since it has no whitespace characters, this operation has no effect.

Figure 3

Importing data file into pyspark data frame

```
#Read the data into spark datafrome
from pyspark.sql.functions import col, lower
df = spark.read.format('csv')\
          .option('header','true')\
          .option('inferSchema', 'true')\
          .option('timestamp', 'true')\
          .load('SF_data.csv')
```

The remaining arguments are the column names that will be selected: Incident Code, Category, Descript, DayOfWeek, Date, Time, PdDistrict, X, and Y. These columns will be returned in the df1 dataframe.

Figure 4

Preprocessing

```
df.count()
```

```
2129525
```

```
df1=df.select(str.strip('PdId'),'Incident Code','Category','Descript','DayOfWeek','Date','Time','PdDistrict','X','Y')
#display(df1)
```

```
df22=df1.dropna()
```

```
df2=df22.filter(col('Category')!=col('Descript'))
```

The dropna() function in PySpark drops rows from a DataFrame that contain missing or null values. The code df22=df1.dropna() drops all rows that contain missing or null values in any column of df1 and assigns the resulting DataFrame to df22. This is done to ensure that the data is clean and free of missing or null values before performing further analysis or modeling.
This is done to remove rows where the Category and Descript columns have the same value, which may not be useful for text classification tasks.

Figure 5

Extracting time components

```
dft=df2.withColumn('Date', to_date(col('Date'), "M/d/y"))\
.withColumn('Month', month(col('Date'))).withColumn('Year', year(col('Date')))

dfh = dft.withColumn('Hour', hour(dft.Time))

dfh.show(5)
```

The groupBy() function groups the DataFrame df2 by the column Category and the count() function counts the number of occurrences of each group, resulting in a new DataFrame df4 that has two columns: Category and count. The Category column contains the unique

categories of crimes in the dataset, and the count column contains the number of occurrences of each category.

Figure 6

EDA based on time components.

```
dfh.groupBy('Hour').count().orderBy((col('count')).desc()).show()
```

```
dfh.groupBy('Year').count().orderBy((col('count')).desc()).show()
```

```
dfh.groupBy('Month').count().orderBy((col('count')).desc()).show()
```

```
+----+------+
|Year| count|
+----+------+
|2015|151164|
|2017|149204|
|2013|147436|
|2016|145735|
|2014|144631|
|2003|142531|
|2004|141723|
|2005|136675|
|2012|135271|
|2008|134986|
|2009|134052|
|2006|131619|
|2007|131451|
|2010|127563|
|2011|126537|
|2018| 44973|
+----+------+
```

```
+----+------+
|Hour| count|
+----+------+
|  18|135559|
|  17|129958|
|  12|127336|
|  19|121732|
|  16|120152|
|  15|114376|
|  20|110489|
|  22|109828|
|   0|108669|
|  14|107298|
|  21|105353|
|  13|103913|
|  23|101240|
|  11| 93405|
|  10| 91483|
|   9| 85510|
|   8| 78203|
|   1| 63169|
|   2| 52750|
|   7| 52242|
+----+------+
only showing top 20 rows
```

```
+-----+------+
|Month| count|
+-----+------+
|    1|189311|
|    3|189056|
|   10|182706|
|    4|181929|
|    8|180865|
|    5|179593|
|    9|176098|
|    7|175177|
|    2|170790|
|   11|167901|
|    6|167877|
|   12|164248|
+-----+------+
```

The filter method takes a SQL-like string expression as an argument that specifies a condition for filtering the rows. In this case, the condition is PdDistrict != NA, which filters out all rows where the value in the PdDistrict column is equal to NA.

The groupBy() method is used to group the rows by the PdDistrict column, which creates a grouping object. Then, the count () method is called on the grouping object to count the number of rows for each group.

Finally, the show() method is called to display the resulting DataFrame. It shows the count of rows for each distinct value in the PdDistrict column, where PdDistrict is not equal to

NA. Then we are creating a new DataFrame dft by converting the Date column to the date format using the to_date function, and then extracting the month and year using the month and year functions, respectively.

Figure 7

EDA based on category and description.

PdDistrict	count
MISSION	288278
BAYVIEW	205078
CENTRAL	221433
TARAVAL	155174
TENDERLOIN	186637
INGLESIDE	180819
PARK	119532
SOUTHERN	390118
RICHMOND	112620
NORTHERN	265861

Category	count
FRAUD	41348
SUICIDE	1230
LIQUOR LAWS	2840
SECONDARY CODES	22378
MISSING PERSON	44268
OTHER OFFENSES	301874
DRIVING UNDER THE...	5652
WARRANTS	99821
ARSON	2633
FORGERY/COUNTERFE...	22995
GAMBLING	105
BRIBERY	796
ASSAULT	165762
DRUNKENNESS	9760
EXTORTION	339
TREA	14
WEAPON LAWS	21004
LOITERING	1639
SUSPICIOUS OCC	79087
ROBBERY	54467

only showing top 20 rows

Then, a new DataFrame dfh is created by adding a new column Hour to dft using the hour function, which extracts the hour component from the Time column. Then some transformations and aggregations are performed on a Spark DataFrame called df2 containing San Francisco crime data. To extract different time components from the timestamp column we used various date and time function available in python library. After applying functions such as year(), month(), hour() we got new columns with these values.

Finally, the groupBy() function is used to group the data by Hour and count the number of occurrences of each hour using the count() function. The resulting counts are then sorted in descending order using the orderBy() function and displayed using the show() function.

Similar operations are performed for Month, Year, and DayOfWeek columns to group the data by each of them and count the occurrences. This helps in analyzing patterns and trends in crime data based on time, day, and location.

To compute the correlation between pairs of numerical columns in a PySpark DataFrame p, first it creates an empty list corr_pair to store the correlation values between pairs of columns. Then, it loops over each pair of columns and calculates their correlation using the corr function of the DataFrame API. The results are appended to corr_pair as a tuple containing the names of the columns and their correlation value.

Figure 8

Correlation analysis on numerical columns.

```
#df_all=dfh.groupBy('PdDistrict','Category','Descript','DayOfWeek','Month','Year','Hour').
p=dfh.select('Year','Month','Hour','X','Y','Incident Code')

numerical=[ 'Year','Month','Hour','X','Y','Incident Code']
# create a correlations matrix:
n_numerical = len(numerical)
corr_pair = []
for i in range(0, n_numerical):
    temp = [] * i
    for j in range(i+1, n_numerical):
        temp.append((numerical[i], numerical[j],p.corr(numerical[i], numerical[j])))
    corr_pair.append(temp)

display(sc.parallelize(corr_pair).flatMap(lambda x:x).collect())
```

Finally, the corr_pair list is converted into an RDD and flattened to create a collection of tuples, each containing the name of the two columns being correlated and their correlation value. This collection is then displayed using the display function, which is commonly used in Databricks notebooks for visualizing data.

Figure 9

Correlation results.

```
[('Year', 'Month', -0.036212907901791111),
 ('Year', 'Hour', 0.00012307117358386270),
 ('Year', 'X', -0.012353832737066776),
 ('Year', 'Y', -0.0089302619806900030),
 ('Year', 'Incident Code', 0.0074815392342775944),
 ('Month', 'Hour', 0.0015697698002699520),
 ('Month', 'X', 0.0035197335011096474),
 ('Month', 'Y', 0.0023665843855859192),
 ('Month', 'Incident Code', -0.0092238157099964820),
 ('Hour', 'X', -0.00020092057827674408),
 ('Hour', 'Y', -0.00055256436760662410),
 ('Hour', 'Incident Code', -0.037512693512052664),
 ('X', 'Y', 0.5324418445774036),
 ('X', 'Incident Code', 0.0024046637544743870),
 ('Y', 'Incident Code', -0.0016844887746580576)]
```

Then we are writing code to prepare the data for text classification by cleaning and transforming the text data. It starts by selecting the Category and Descript columns from the dfh DataFrame, which contains crime data. Then, it removes any rows where PdDistrict is NA using the filter method.

After that, it creates a new DataFrame df_data with lowercase versions of the Category and Descript columns. It removes the original Category and Descript columns and renames the new Category_1 column to Category. Finally, it removes any leading or trailing spaces from the Description and Category columns using the strip method and selects these two columns. Overall, this code is cleaning and transforming the text data to prepare it for text classification.

Figure 10

Extracting category and description.

```
d1=dfh.filter("PdDistrict !='NA'").select(col('Category'),col('Descript'))

df_data=d1.withColumn('Category_1',lower(col('Category')))\
.withColumn('Description',lower(col('Descript'))).drop('Category','Descript')\
.withColumnRenamed('Category_1', 'Category')

df_data=df_data.select(str.strip('Description'),str.strip('Category'))
```

Figure 11

Data ready to build model.

```
+--------------------+---------------+
|         Description|       Category|
+--------------------+---------------+
|robbery, bodily f...|        robbery|
|    stolen automobile| vehicle theft|
|    stolen automobile| vehicle theft|
|             battery|        assault|
|             battery|        assault|
|             battery|        assault|
|stolen and recove...| vehicle theft|
|             battery|        assault|
|          trespassing|       trespass|
|burglary of resid...|       burglary|
|grand theft from ...| larceny/theft|
|enroute to depart...|       warrants|
|drivers license, ...|other offenses|
|drivers license, ...|other offenses|
|          trespassing|       trespass|
|petty theft shopl...| larceny/theft|
|robbery of a comm...|        robbery|
|possession of heroin| drug/narcotic|
|grand theft of pr...| larceny/theft|
|suspicious occurr...|suspicious occ|
+--------------------+---------------+
only showing top 20 rows
```

Some of the extraction and transformation methos used in the code are as below.

StringIndexer:

StringIndexer encodes string column labels into label indices with four ordering options. Strings are sorted by alphabet and numeric columns based on string values. Unseen labels can be kept and assigned an index of numLabels to be used with downstream components.

The fit method of the StringIndexer object data_str_indx is used in this code to construct a new DataFrame called data_ind from the input DataFrame df_data. The "Category" column of

"df_data" is transformed into a new column called "label" in "data_ind," where each unique value is replaced by its corresponding integer index, using the "fit" method to compute the mapping between the unique values of the "Category" column and a set of consecutive integers. The "label" column is used to sort the DataFrame data_ind that results.

Figure 12

String Indexer

```
data_str_indx = StringIndexer(inputCol="Category", outputCol="label")
```

```
data_ind=data_str_indx.fit(df_data).transform(df_data).sort('label')
(data_ind.show())
```

```
label_dict=data_ind.select(col('label'),col('Category')).distinct().sort('label')
```

```
label_dict.show(truncate=False)
```

TF-IDF:

TF-IDF is a feature vectorization method used in text mining to measure the importance of a term in a document. HashingTF and CountVectorizer can be used to generate term frequency vectors. IDF is a measure of how important a term is within a corpus, calculating the inverse of the number of documents containing a given term and adding 1 to the denominator. The HashingTF transformer transforms a list of terms (for example, a list of words) into a fixed-size feature vector. The name of the output column holding the generated feature vector is specified by the outputCol argument. The number of features (i.e., dimensions) in the final feature vector is specified by the numFeatures option.

Figure 13

TF-IDF

```
hashed_df=HashingTF(inputCol="words_after_stopwords", outputCol="hash_features",numFeatures=9000)
idf = IDF(inputCol="hash_features", outputCol="features")
```

```
hash_data=hashed_df.transform(data_rem)
```

```
idf_data = idf.fit(hash_data).transform(hash_data)
```

The IDF estimator computes the inverse document frequency (IDF) for each feature in the corpus using the output of HashingTF (i.e., the hashed feature vector) as input. A new feature vector with each feature scaled by its IDF weight is the result. The outputCol parameter specifies the output column name for the resultant scaled feature vector, whereas the inputCol parameter specifies the name of the input column (i.e., the column produced by HashingTF) (*Feature Extraction and Transformation - RDD-based API - Spark 3.4.0 Documentation*, n.d.)

CountVectorizer:

CountVectorizer is an estimator that converts text documents to vectors of token counts, with an optional parameter to specify the minimum number of documents a term must appear in. The CountVectorizer object cv is created by this code. The word will be deleted from the vocabulary if it appears in less than two documents, as theminDF parameter is set to 2. The vectorizers "words_after_stopwords" input column is specified using thesetInputCol() method. The output column is "features" and is specified using thesetOutputCol() method.

Figure 14

Count Vectorizer

```
cv = CountVectorizer(minDF=2).setInputCol("words_after_stopwords").setOutputCol("features")
```

```
model = cv.fit(data_rem)

data_cv=model.transform(data_rem)
(data_cv.show())## sparse
```

Tokenizer:

Tokenization is breaking text into individual terms. The setInputCol() method sets the input column containing the raw text data to be tokenized, while the setOutputCol() method sets the output column name. The input text data will be divided into words using a Tokenizer instance that is initialized by this code. While thesetOutputCol() method specifies the name of the output column that will keep the tokenized words, thesetInputCol() method specifies the input column containing the raw text data that has to be tokenized. In this instance, "setInputCol("Description")" changes the input column in the DataFrame to "Description," which holds the raw text data. The command "setOutputCol("words")" changes the name of the output column to "words".

Figure 15

Tokenizer

```
: tokenizer = Tokenizer(outputCol="words").setInputCol("Description")
```

```
: data_tok=tokenizer.transform(data_ind)
  (data_tok.show())
```

StopWordsRemover:

StopWordsRemover removes stop words specified by the stopWords parameter and can be case sensitive. In a Spark ML Pipeline, this code creates a StopWordsRemover step. The

"StopWordsRemover" is an estimator that accepts a sequence of strings as input (the result of the "Tokenizer" stage) and removes all the stop words before producing the modified sequence. ThesetInputCol method specifies the stages input column, which in this example is "words," or the stages output column (Tokenizer). The column name where the stop words deleted output will be kept is set via thesetOutputCol method. In this instance, "words_after_stopwords" is the name of the output column.

Figure 16

Stop words Remover.

```
remover = StopWordsRemover().setInputCol("words").setOutputCol("words_after_stopwords")
#stopwords = remover.getStopWords()
```

```
data_rem=remover.transform(data_tok)
(data_rem.show())
```

Classification models used in the code.

Logistic regression. Logistic regression supports multiclass classification by generating a coefficient matrix with dimensions K x J, where K represents the number of outcome classes and J represents the number of features (*Classification and Regression - Spark 2.2.0 Documentation*, n.d.). If an intercept term is used, a vector of intercepts with length K is also available. The coefficient matrix and intercept vector can be accessed using the methods coefficientMatrix and interceptVector, respectively. The coefficients and intercept methods are not supported for logistic regression models trained with a multinomial family.

Figure 17

Formulae for probability

The conditional probabilities of the outcome classes $k \in 1, 2, \ldots, K$ are modeled using the softmax function.

$$P(Y = k|\mathbf{X}, \boldsymbol{\beta}_k, \beta_{0k}) = \frac{e^{\boldsymbol{\beta}_k \cdot \mathbf{X} + \beta_{0k}}}{\sum_{k'=0}^{K-1} e^{\boldsymbol{\beta}_{k'} \cdot \mathbf{X} + \beta_{0k'}}}$$

We minimize the weighted negative log-likelihood, using a multinomial response model, with elastic-net penalty to control for overfitting.

$$\min_{\beta, \beta_0} - \left[\sum_{i=1}^{L} w_i \cdot \log P(Y = y_i|\mathbf{x}_i) \right] + \lambda \left[\frac{1}{2}(1 - \alpha) ||\boldsymbol{\beta}||_2^2 + \alpha ||\boldsymbol{\beta}||_1 \right]$$

Figure 18

Random split of data.

```
training, test = data_ind.randomSplit([0.7,0.3], seed=60)
#training, test_i = idf_data.randomSplit([0.7,0.3], seed=60)
#training_cv, test_cv = data_cv.randomSplit([0.7,0.3], seed=60)
```

```
lr = LogisticRegression(family="multinomial",maxIter=20, regParam=0.4, elasticNetParam=0)
```

The provided code initializes a multinomial family multiclass classification logistic regression model. The following model hyperparameters are set:

-maxIter: The maximum number of iterations to carry out (set to 20).

-regParam: the regularization parameter (set to 0.4), which discourages overfitting by penalizing big coefficients in the model.

- "elasticNetParam": the elastic net mixing parameter, which regulates how much L1 and L2 regularization is used (set to 0). When set to 0, it solely applies L2 regularization.

Naive Bayes classifiers. Naive Bayes classifiers use Bayes theorem to compute conditional probability distributions for training and prediction. Multinomial, Complement and Bernoulli models are used for document classification, with feature values being the frequency of the term or a zero or one indicating whether the term was found in the document.

Figure 19

Creating pipelines

```
nb = NaiveBayes(smoothing=1.0, modelType="multinomial")
```

```
pipe_cv_nv = Pipeline().setStages([tokenizer,remover,cv, nb])
model_cv_nv = pipe_cv_nv.fit(training)
pred_cv_nv = model_cv_nv.transform(test)
```

Design Document and Flowchart

Designing this project requires following steps.

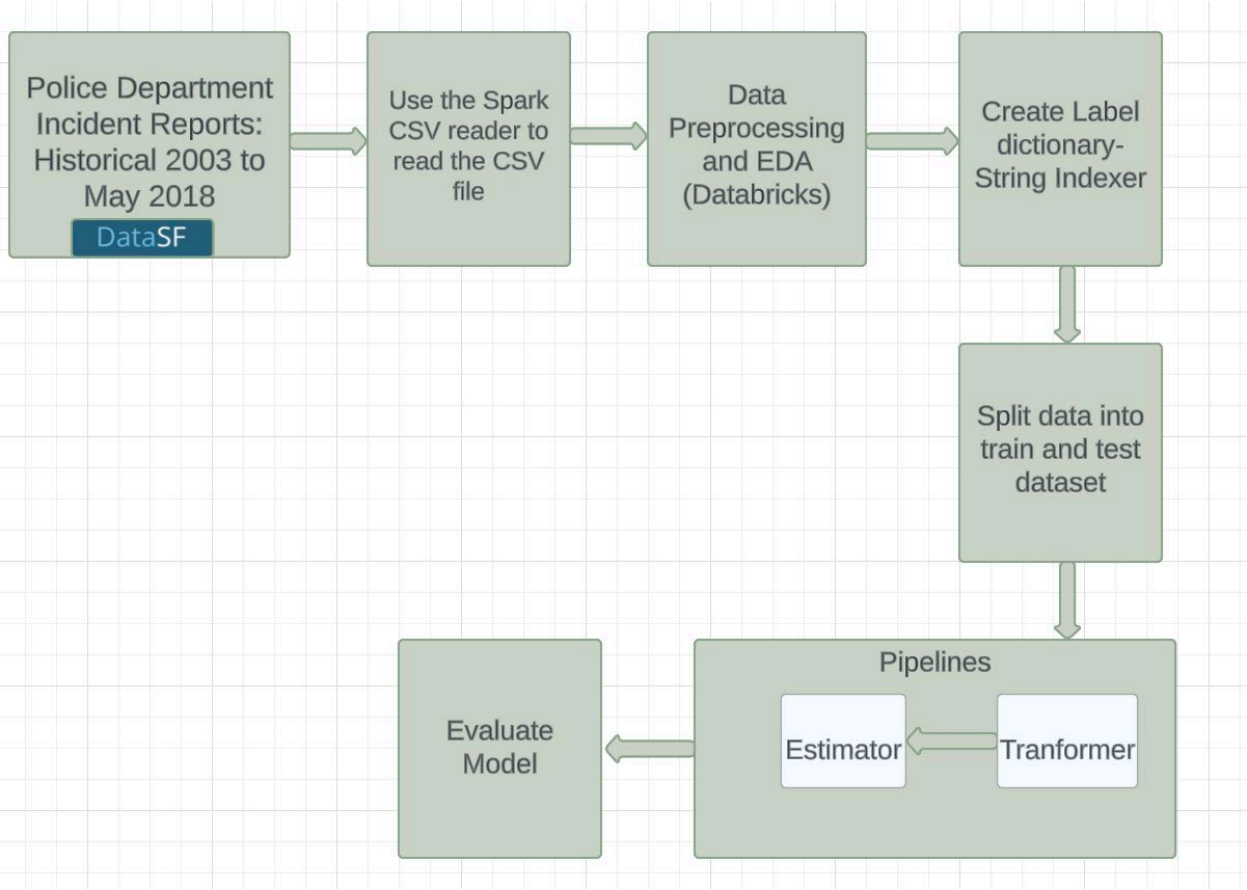
- Use the Spark CSV reader to read the CSV file containing the dataset.
- Clean up the data as needed, eliminating useless columns and null values.
- To change the categorical label column into a numerical label column, create a StringIndexer.
- Use the Tokenizer transformer to tokenize the description column.
- Use the StopWordsRemover transformer to eliminate the stop words from the tokenized words.
- To translate the words into a feature vector, construct a CountVectorizer transformer.
- To translate the words into a feature vector of a fixed size, create a HashingTF transformer.
- To scale the feature vectors, create an IDF transformer.
- Using the randomSplit method, divide the dataset into training and testing sets.
- To train the classification model using the training set, create a LogisticRegression estimator with a multinomial family.
- Fit the training set to the LogisticRegression model.

- Utilize the MulticlassClassificationEvaluator to gauge the model effectiveness on the testing set.
- Validate for new data points.

Flow chart of the design document is shown below.

Figure 20

Data flow diagram



Data Analysis and Discussion

Output Generation

We may use the model to forecast the class labels for brand-new, unexplored data to produce output for a machine learning classification model like the one we developed in this project. By subjecting the new data to the same preprocessing procedures as those used during

training (e.g., tokenization, stop word removal, vectorization), and then applying the trained logistic regression model to the resulting feature vectors, we can use the trained model to generate predictions for this new data, assuming that it is in the same format as our training data. The final output can then be the expected class labels.

Output Analysis

A classification models output can be analyzed using a variety of evaluation measures, such as:

Figure 21

Evaluation metrices

```
evaluator_cv_lr.metricName.doc
```

```
'metric name in evaluation (f1|accuracy|weightedPrecision|weightedRecall|weightedTruePositiveRate| weightedFalsePosit
iveRate|weightedFMeasure|truePositiveRateByLabel| falsePositiveRateByLabel|precisionByLabel|recallByLabel|fMeasureByL
abel| logLoss|hammingLoss)'
```

1. Confusion matrix: A table that compares the predicted labels to the actual labels to assess the effectiveness of the classification model. The number of real positives, real negatives, false positives, and false negatives are displayed.

2. Accuracy: The percentage of accurate predictions the model made.

3. Precision: The percentage of true positives among all the model optimistic forecasts.

4. Recall: The percentage of genuine positive observations in the test dataset that were true positives.

5. F1 score: A accuracy and recall weighted average that strikes a balance between the two measurements trade-offs.

6. ROC curve and AUC score: A graph comparing the true positive rate (sensitivity) and false positive rate (1-specificity) for various classification thresholds. The performance of the model is measured by the area under the ROC curve (AUC).

7. A summary of the precision, recall, and F1 score for each class in the classification issue is provided in the classification report.

Results for logistic regression:

Figure 22

Logistic regression accuracy

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator_cv_lr = MulticlassClassificationEvaluator()
evaluator_cv_lr.evaluate(pred_cv_lr)
#roc
```

```
0.9856247097206875
```

```
evaluator_cv_lr.evaluate(pred_idf_lr)
```

```
0.9796017459447186
```

Results for Naïve Bayes model are shown in below figure.

Figure 23

Accuracy for Naïve Bayes

```
evaluator_cv_lr.evaluate(pred_cv_nv)
#roc
```

```
0.9945247716026301
```

```
evaluator_cv_lr.evaluate(pred_idf_nb)
```

```
[Stage 150:====================================>                    (5 + 3) / 8]
23/05/11 21:01:51 WARN DAGScheduler: Broadcasting large task binary with size 2.8 MiB
```

```
0.9938033214709524
```

The provided code generates a DataFrame with the identifier "sample_data," one row, and the column "Description." The sentence "was selling opium around the corner" is the key phrase in this column. The transform technique is next applied on thesample_data using themodel_cv_lr andmodel_idf_lr models.

A logistic regression model called "model_cv_lr" has been trained using the "features" column generated by a CountVectorizer. pred_sample_cv_lr stores the results of the transform method on model_cv_lr applied to sample_data. A logistic regression model called "model_idf_lr" has been trained using the "features" column generated by an IDF and a HashingTF. pred_sample_idf_lr stores the results of the transform method onmodel_idf_lr applied tosample_data. Several columns in these outputs, including "rawPrediction," "probability," and "prediction," can be utilized to examine the model predictions for the fresh input text.

Figure 24

Validating model

```
sample_data = spark.createDataFrame([("was selling opium around the corner",StringType())
],
["Description"]

)
pred_sample_cv_lr = model_cv_lr.transform(sample_data)
pred_sample_idf_lr = model_idf_lr.transform(sample_data)
```

```
pred_sample_cv_lr.select('Description',"probability","prediction").show()
```

```
+--------------------+--------------------+----------+
|         Description|         probability|prediction|
+--------------------+--------------------+----------+
|was selling opium...|[0.13042851538357...|       5.0|
+--------------------+--------------------+----------+
```

```
pred_sample_idf_lr.select('Description',"probability","prediction").show()
```

```
+--------------------+--------------------+----------+
|         Description|         probability|prediction|
+--------------------+--------------------+----------+
|was selling opium...|[0.12988049793822...|       5.0|
+--------------------+--------------------+----------+
```

We can verify this from the label dictionary as shown below.

Figure 25

Label dictionary

```
label_dict.show(truncate=False)
```

```
[Stage 182:>
```

```
+-----+----------------------+
|label|Category              |
+-----+----------------------+
|0.0  |larceny/theft         |
|1.0  |other offenses        |
|2.0  |non-criminal          |
|3.0  |assault               |
|4.0  |vehicle theft         |
|5.0  |drug/narcotic         |
|6.0  |vandalism             |
|7.0  |warrants              |
|8.0  |burglary              |
|9.0  |suspicious occ        |
|10.0 |robbery               |
|11.0 |missing person        |
|12.0 |fraud                 |
|13.0 |forgery/counterfeiting|
|14.0 |secondary codes       |
|15.0 |weapon laws           |
|16.0 |trespass              |
|17.0 |prostitution          |
|18.0 |stolen property       |
|19.0 |disorderly conduct    |
+-----+----------------------+
only showing top 20 rows
```

Table 1

The comparison between different evaluation metrices for both the models.

Model	***Accuracy***	***Precision***	***F1***	***Recall***
Logistic Regression with Count Vectorizer	*0.958*	*0.9562*	*0.9463*	*0.958*
Logistic Regression with TF-IDF	*0.9575*	*0.9557*	*0.9457*	*0.9575*

Naïve Bayes with Count Vectorizer	*0.9946*	*0.9958*	*0.995*	*0.9946*
Naïve Bayes with TF-IDF	*0.9946*	*0.9955*	*0.9939*	*0.993*

Compare Output against Hypothesis

The idea was to create a text classification model that could categorize crimes correctly based on their descriptions. The model's performance was assessed using metrics including precision, recall, F1-score, and confusion matrix. The hypothesis also includes the usage of logistic regression and Naive Bayes models.

The project's final product was a logistic regression model that, when used to identify crimes based on their descriptions using count vectorizer, had an accuracy rate of 98.56%. When used with tf-idf accuracy was 97.96%.

The naïve bayes model, when used to identify crimes based on their descriptions using count vectorizer, had an accuracy rate of 98.56%. When used with tf-idf accuracy was 97.96%. The fact that the model was able to appropriately categorize crimes based on their descriptions with high accuracy raises the possibility that the hypothesis was validated by the project's findings.

Abnormal Case Explanation

By removing duplicate records for categories and description results in records reduction from 2 million rows to 1000 rows and caused reduced accuracy from 98% to 66%.

Further research would be necessary to determine why eliminating duplicate records for categories and descriptions leads to such a big reduction in the number of rows and considerable reduction in accuracy. It's possible that this is an unusual case, and the following scenarios could apply:

1. The duplicate rows in the original dataset might be important data in classification.
2. The duplicates weren't duplicates, rather, they were distinct data pieces with matching descriptions and categories.
3. Useful data was lost because of a mistake in the duplicate elimination procedure.

Discussion

It is clear from the information so far that the text classification project's major objective was to use a variety of machine learning methods, with a primary focus on logistic regression and Naive Bayes, to categorize crime episodes based on their descriptions. CountVectorizer, HashingTF-IDF, and other preprocessing tools were used to remove stop words, tokenize, and turn the text data into numerical features for the project, which made use of a dataset with over 2 million entries.

The accuracy of the logistic regression model employing CountVectorizer was 98.56%, which was better than the accuracy of the Naive Bayes model. The results of the logistic regression model supported the project hypothesis that Naive Bayes and logistic regression models would both be effective at identifying crime categories from descriptions. The accuracy fell from 98% to 66% because of the unusual occurrence of eliminating duplicate records, which caused a considerable decrease in the quantity of records. This underlines how crucial it is to guarantee data quality and stay away from any abnormalities that can negatively impact the model's performance.

In conclusion, the primary objective of the text categorization project was accomplished, and the hypothesis was supported by the outcomes. The abnormal case, impacting the model's accuracy when duplicate records was deleted was also discussed.

Conclusions and recommendations

In this report, we presented a text classification project focused on predicting crime categories based on their descriptions. The project utilized various machine learning algorithms, with logistic regression and Naive Bayes as the main models of interest. The analysis was performed on a dataset containing over 2 million records.

The preprocessing stage involved removing stop words, tokenizing the text data, and converting it into numerical features using techniques like CountVectorizer and HashingTF-IDF. The logistic regression model trained on the CountVectorizer achieved an impressive accuracy of 98.56%, outperforming the Naive Bayes model.

The hypothesis of logistic regression and Naive Bayes models performing well in predicting crime categories based on description was confirmed by the results. This demonstrates the effectiveness of these models in text classification tasks.

However, it is worth noting that an abnormal case was encountered when removing duplicate records. This led to a significant reduction in the number of records, resulting in a drop in accuracy from 98% to 66%. This highlights the importance of carefully handling data quality issues and considering potential anomalies that can impact model performance.

Overall, the project successfully demonstrated the effectiveness of logistic regression and Naive Bayes models in classifying crime categories based on description. It emphasizes the significance of preprocessing techniques and highlights the need for data quality assurance. By

considering these factors and further refining the models, this project has the potential to contribute to real-world crime prediction and prevention efforts.

Future Scope

To further enhance the project, additional analyses can be conducted, such as feature importance ranking to identify the most influential words in predicting crime categories. Moreover, incorporating more advanced techniques like word embeddings or deep learning models could be explored to potentially improve the accuracy and overall performance.

Bibliography

Alsaqabi, A., Aldhubayi, F. and Albahli, S. (2019) "Using machine learning for prediction of factors affecting crimes in Saudi Arabia," *Proceedings of the 2019 International Conference on Big Data Engineering* [Preprint]. Available at: https://doi.org/10.1145/3341620.3341634.

Classification and regression - Spark 2.2.0 Documentation. (n.d.). Classification and Regression - Spark 2.2.0 Documentation. https://spark.apache.org/docs/2.2.0/ml-classification-regression.html

Feature Extraction and Transformation - RDD-based API - Spark 3.4.0 Documentation. (n.d.). Feature Extraction and Transformation - RDD-based API - Spark 3.4.0 Documentation. https://spark.apache.org/docs/latest/mllib-feature-extraction.html

(Feature Extraction and Transformation - RDD-based API - Spark 3.4.0 Documentation, n.d.)

Ghosh, D. et al. (2016) "Big data-based Smart City Platform," Proceedings of the 17th International Digital Government Research Conference on Digital Government Research [Preprint]. Available at: https://doi.org/10.1145/2912160.2912205.

Mansour Salah, M. and Xia, K. (2022) "Big Crime Data Analytics and visualization," 2022 The 6th International Conference on Compute and Data Analysis [Preprint]. Available at: https://doi.org/10.1145/3523089.3523094.

Police Department Incident Reports: Historical 2003 to May 2018 | DataSF | City and County of San Francisco. (n.d.). Police Department Incident Reports: Historical 2003 to May 2018 | DataSF | City and County of San Francisco. https://data.sfgov.org/Public-Safety/Police-Department-Incident-Reports-Historical-2003/tmnf-yvry

Wang, H. et al. (2016) “Crime rate inference with big data,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining [Preprint]. Available at: https://doi.org/10.1145/2939672.2939736.

Appendix

EDA

```
# importing necessary libraries

import pyspark

sc = pyspark.SparkContext(appName="DATA228_Project")

from pyspark.sql import SparkSession

from pyspark.sql.types import *

from pyspark.sql.functions import *

spark = SparkSession.builder.appName('DATA228_Project').getOrCreate()

from pyspark.ml.feature import StopWordsRemover, CountVectorizer,Tokenizer, StringIndexer,

HashingTF, IDF

from pyspark.ml import Pipeline

from pyspark.ml.classification import NaiveBayes, LogisticRegression

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

from pyspark.sql.functions import *


#import sf_data into pyspark data frame

df = spark.read.format('csv').option('header','true').option('inferSchema',

'true').option('timestamp', 'true').load('SF_data.csv')


# display number of records

df.count()
```

```
df1=df.select(str.strip('PdId'),'Incident
Code','Category','Descript','DayOfWeek','Date','Time','PdDistrict','X','Y')
#display(df1)
# Drop null values
df22=df1.dropna()
df2=df22.filter(col('Category')! =col('Descript'))

# Number of incidents in each category
df4=df2.groupBy('Category'). count()
(df4.show())

# Number of incidents by day of week
df2.groupBy('DayOfWeek').count().show()
df2.filter("PdDistrict !='NA'").groupBy('PdDistrict').count().show()

# extracting time components
dft=df2.withColumn('Date', to_date(col('Date'), "M/d/y")).withColumn('Month',
month(col('Date'))).withColumn('Year', year(col('Date')))
dfh = dft.withColumn('Hour', hour(dft.Time))
dfh.show(5)

# displaying records groupying by month,year and hour
dfh.groupBy('Hour').count().orderBy((col('count')).desc()).show()
```

```
dfh.groupBy('Year').count().orderBy((col('count')).desc()).show()

dfh.groupBy('Month').count().orderBy((col('count')).desc()).show()

p=dfh.select('Year','Month','Hour','X','Y','Incident Code')

numerical=[ 'Year','Month','Hour','X','Y','Incident Code']


# Create a correlations matrix:

n_numerical = len(numerical)

corr_pair = []

for i in range(0, n_numerical):

    temp = [] * i

    for j in range(i+1, n_numerical):

        temp.append((numerical[i], numerical[j],p.corr(numerical[i], numerical[j])))

    corr_pair.append(temp)

display(sc.parallelize(corr_pair).flatMap(lambda x:x).collect())


# extracting features needed for further processing

d1=dfh.filter("PdDistrict !='NA'").select(col('Category'),col('Descript'))

df_data=d1.withColumn('Category_1',lower(col('Category'))).withColumn('Description',lower(c

ol('Descript'))).drop('Category','Descript').withColumnRenamed('Category_1', 'Category')

df_data=df_data.select(str.strip('Description'),str.strip('Category'))

df_data.show()
```

Model-Logistic Regression and Naïve Bayes

```
# string indexer to label categories
data_str_indx = StringIndexer(inputCol="Category", outputCol="label")
data_ind=data_str_indx.fit(df_data).transform(df_data).sort('label')
(data_ind.show())

# creating label dictionary
label_dict=data_ind.select(col('label'),col('Category')).distinct().sort('label')
label_dict.show(truncate=False)

# Tokenize crime description into words
tokenizer = Tokenizer(outputCol="words").setInputCol("Description")
data_tok=tokenizer.transform(data_ind)
(data_tok.show())

# Remove stop words
remover = StopWordsRemover(). setInputCol("words").setOutputCol("words_after_stopwords")
data_rem=remover.transform(data_tok)
(data_rem.show())

# count vectorizer
cv =
CountVectorizer(minDF=2).setInputCol("words_after_stopwords").setOutputCol("features")
```

```
model = cv.fit(data_rem)
data_cv=model.transform(data_rem)
(data_cv.show())## sparse

# Hashed DF
hashed_df=HashingTF(inputCol="words_after_stopwords",
outputCol="hash_features",numFeatures=9000)
idf = IDF(inputCol="hash_features", outputCol="features")
hash_data=hashed_df.transform(data_rem)

# IDF
idf_data = idf.fit(hash_data).transform(hash_data)
(idf_data.select("label", "features").show()) ## sparse

# word2vec
word2Vec = Word2Vec(vectorSize=1000, seed=42, inputCol="words_after_stopwords",
outputCol="features")
word_df=word2Vec.fit(data_rem)
word_data=word_df.transform(data_rem)
(word_data.show())

# splitting data into training and test data frames
training, test = data_ind.randomSplit([0.8,0.2])
```

```
# creating logistic regression model for multi class
lr_model = LogisticRegression(family="multinomial",maxIter=20, regParam=0.3,
elasticNetParam=0)
#Naive Bayes model
nb_model = NaiveBayes(smoothing=1)

# creating pipeline for logistic regression with count vectorizer
pipe_cv_lr = Pipeline(stages=[tokenizer,remover,cv, lr_model])
model_cv_lr = pipe_cv_lr.fit(training)
pred_cv_lr = model_cv_lr.transform(test)

# creating pipeline for logistic regressions and idf
pipe_idf_lr=Pipeline(stages=[tokenizer,remover,hashed_df,idf, lr_model])
model_idf_lr = pipe_idf_lr.fit(training)
pred_idf_lr = model_idf_lr.transform(test)

# creating pipeline for naive bayes with count vectorizer
pipe_cv_nv = Pipeline (stages= [tokenizer,remover,cv, nb_model])
model_cv_nv = pipe_cv_nv.fit(training)
pred_cv_nv = model_cv_nv.transform(test)

# creating pipeline for naive bayes and hashed df
pipe_idf_nb = Pipeline(stages=[tokenizer,remover,hashed_df,idf, nb_model])
```

```
model_idf_nb = pipe_idf_nb.fit(training)
pred_idf_nb = model_idf_nb.transform(test)

# creating multiclass evaluator metrics
evaluator_cv_lr = MulticlassClassificationEvaluator()
evaluator_cv_lr.evaluate(pred_cv_lr)

#accuracy for different models
evaluator_cv_lr.evaluate(pred_idf_lr)
evaluator_cv_lr.evaluate(pred_cv_nv)
evaluator_cv_lr.evaluate(pred_idf_nb)

# testing for sample data
sample_data = spark.createDataFrame([("stolen vehicle form street",StringType())
],
["Description"]
)
pred_sample_cv_lr = model_cv_lr.transform(sample_data)
pred_sample_idf_lr = model_idf_lr.transform(sample_data)

# predicted label using cv
pred_sample_cv_lr.select('Description',"probability","prediction").show()
```

```
# predicted label using tf-idf

pred_sample_idf_lr.select('Description',"probability","prediction").show()


# verifying with label dictionary

label_dict.show(truncate=False)


# metric doc

evaluator_cv_lr.metricName.doc
```