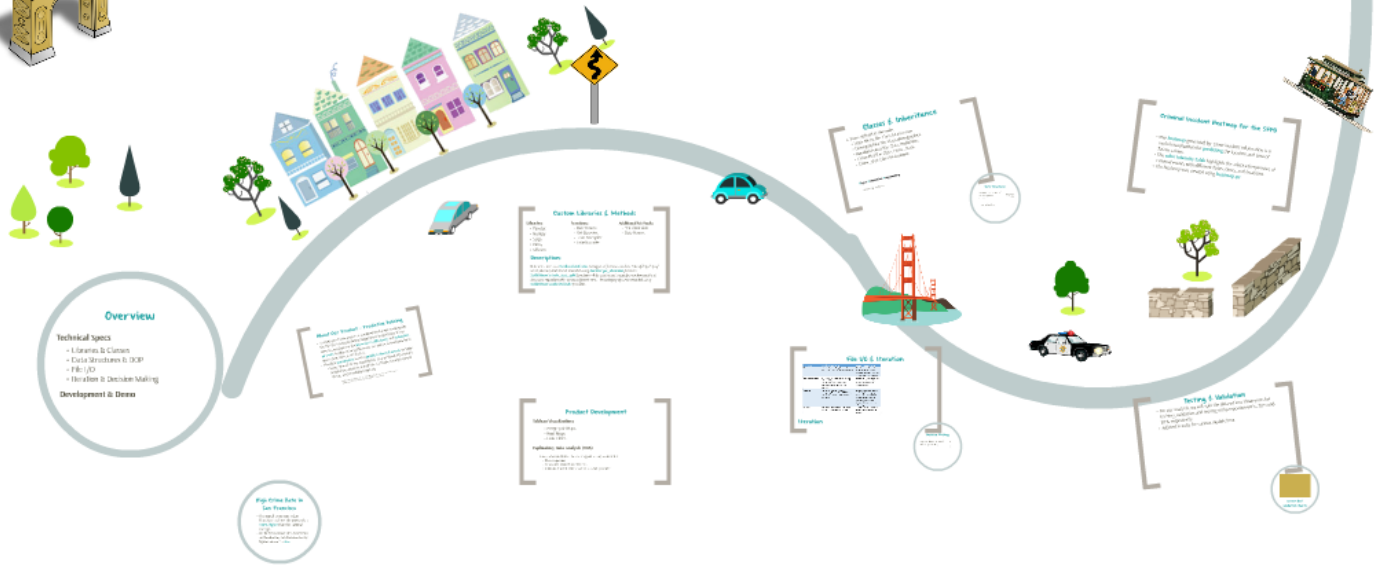
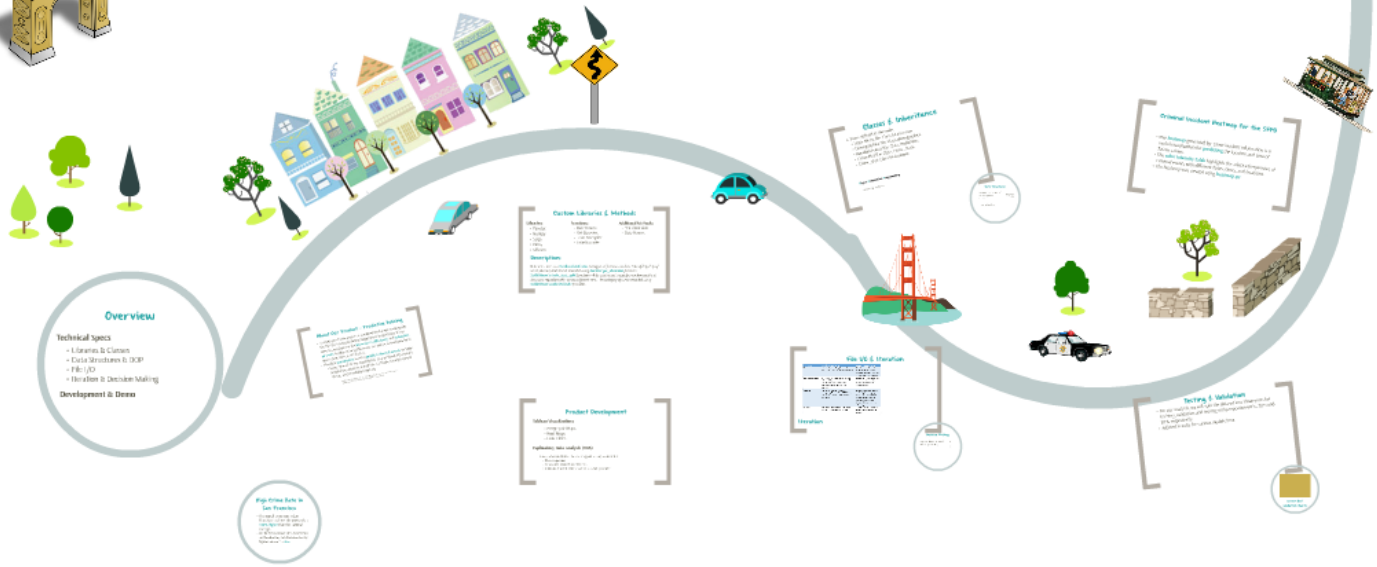
San Francisco Crime Prevention

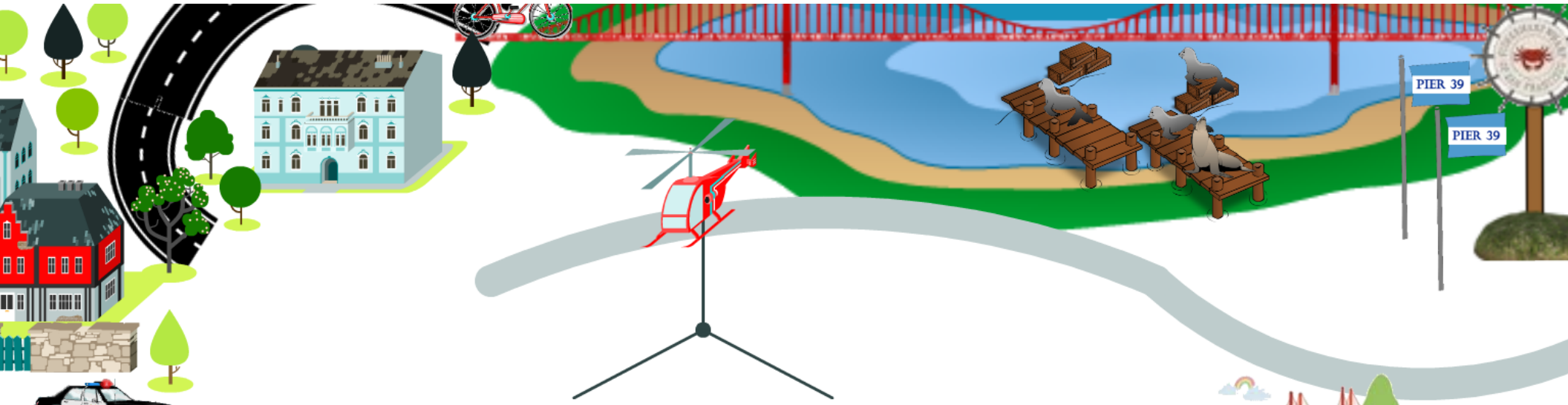
Nicole Dziedic, Siddhant Garg,
Rushil Goyal, Danielle Henriquez



San Francisco Crime Prevention

Nicole Dziedic, Siddhant Garg,
Rushil Goyal, Danielle Henriquez





San Francisco Crime Prevention

Nicole Dziedic, Siddhant Garg,
Rushil Goyal, Danielle Henriquez

Overview

Technical Specs

- Libraries & Classes
- Data Structures & OOP
- File I/O
- Iteration & Decision Making

Development & Demo

High Crime Rate in San Francisco

- The overall crime rate in San Francisco has been documented as **106% higher** than the national average.
- While the national rates have been on the decline, San Francisco's city figures are on the **rise**.

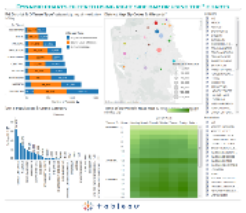
About Our Product - Predictive Policing

- Through predictive analytics, we developed a reference guide for the San Francisco Police Department to predict criminal events and allow for the **increase in efficiency** and **reduction of costs** by determining the necessary police surveillance for a given date, time, and location.
- The data **parameters** used to **predict criminal events** include:
  - Date, Type of Crime, Description, Day of Week, PD District, Resolution, Address, Longitude, Latitude, Unemployment Status, and Population Density.

The dataset includes San Francisco Crime Data from 2003 to 2015 and was retrieved from SF OpenData, the San Francisco Government's Open Data platform.

Product Development

Tableau Visualizations



- Geographic Maps
- Heat Maps
- Data Tables

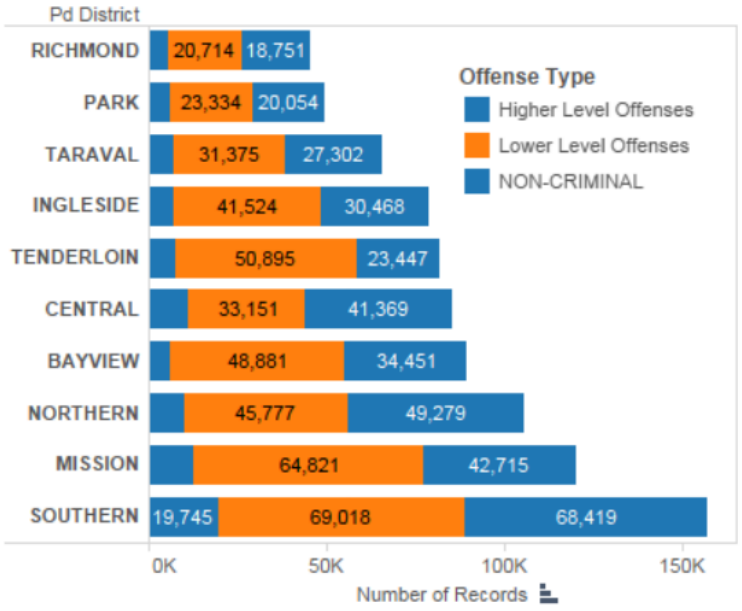
Exploratory Data Analysis (EDA)

Used data visualization tools during preliminary analysis for:

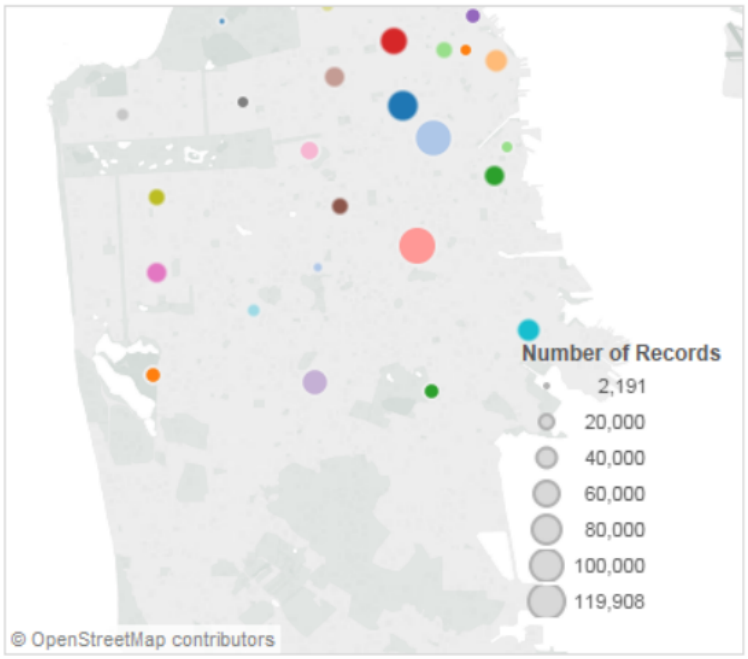
- Data Inspection
- Summarize Main Characteristics
- Minimize the risk of incorrect or misleading results

DYNAMIC CHARTS FILTERED USING RIGHT SIDE BAR OR USING THE * CHARTS

Pd District & Offense Type*-Selected Zip/Day of Week&Time of Day



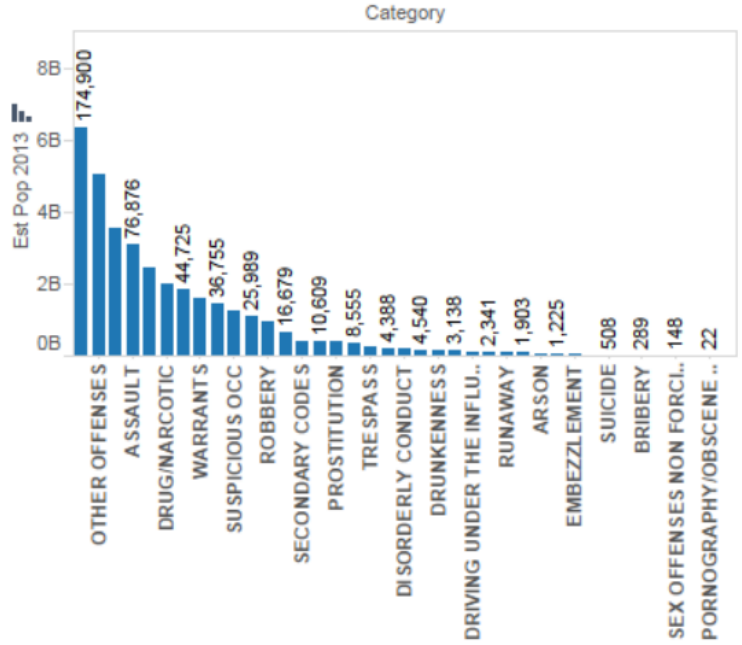
District Map Zip Codes & #Records*



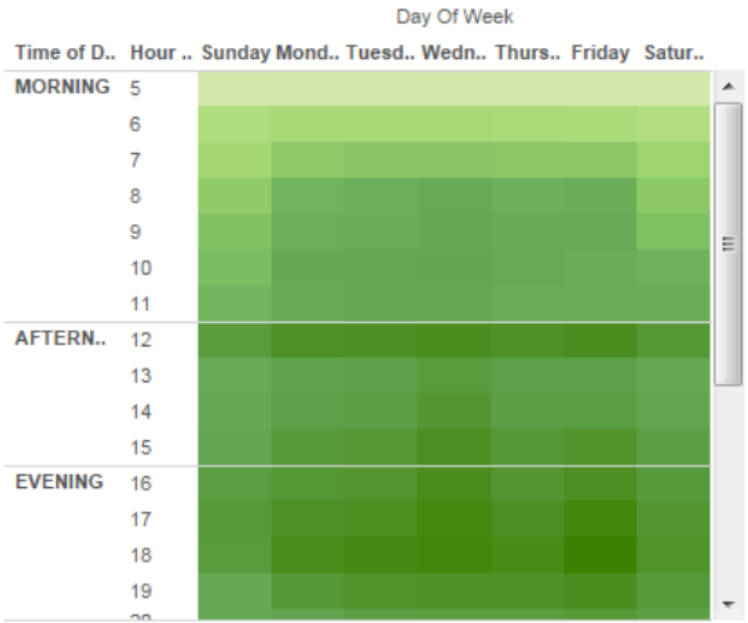
Category

- (All)
- ARSON
- ASSAULT
- BAD CHECKS
- BRIBERY
- BURGLARY
- DISORDERLY C...
- DRIVING UNDER...
- DRUG/NARCOTIC
- DRUNKENNESS
- EMBEZZLEMENT
- EXTORTION
- FAMILY OFFENS...
- FORGERY/COU...
- FRAUD
- GAMBLING
- KIDNAPPING
- LARCENY/THEFT
- LIQUOR LAWS
- LOITERING
- MISSING PERSON
- NON-CRIMINAL
- OTHER OFFENS...

2013 Population & Crime Category



Time of Day/Week Heat Map-By Time of Day for selected PD District, Zip

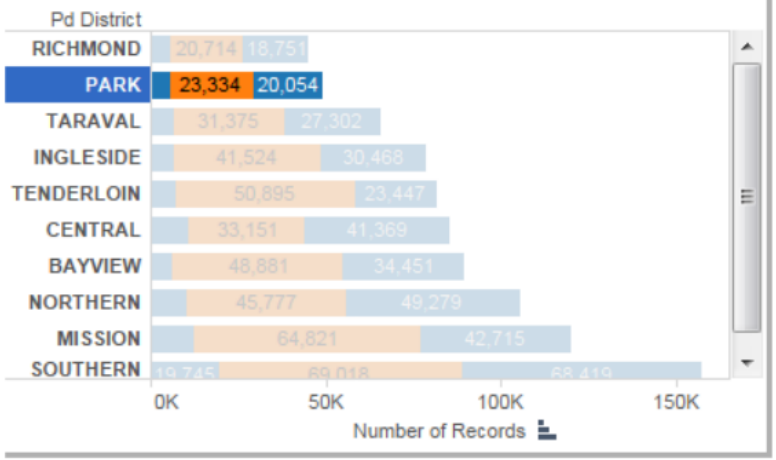


Resolution

- (All)
- ARREST, BOOKED
- ARREST, CITED
- CLEARED-CONTAC...
- COMPLAINANT REF...
- DISTRICT ATTORN...
- EXCEPTIONAL CLE...
- JUVENILE ADMONI...
- JUVENILE BOOKED
- JUVENILE CITED
- JUVENILE DIVERTED
- LOCATED
- NONE
- NOT PROSECUTED
- PROSECUTED BY O...
- PROSECUTED FOR ...
- PSYCHOPATHIC CA...
- UNFOUNDED

Dynamic charts filtered Showing "Park" Crime Stats

Pd District & Offense Type*-Selected Zip/Day of Week&Time of Day

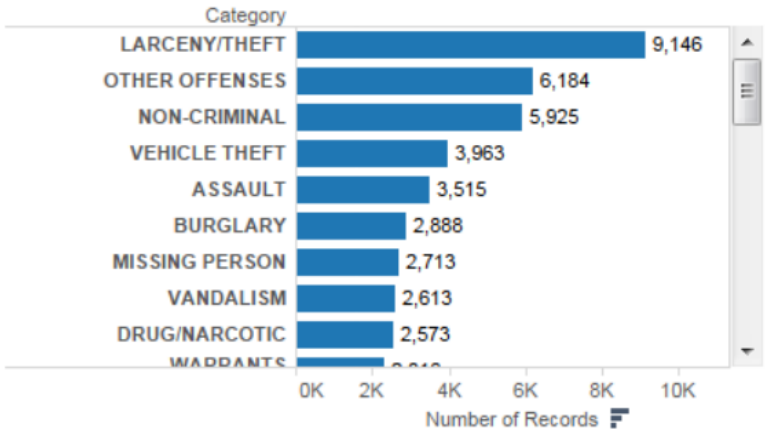


District Demographic Summary - By Zip for selected PD District, Day of Week&Time of Day*

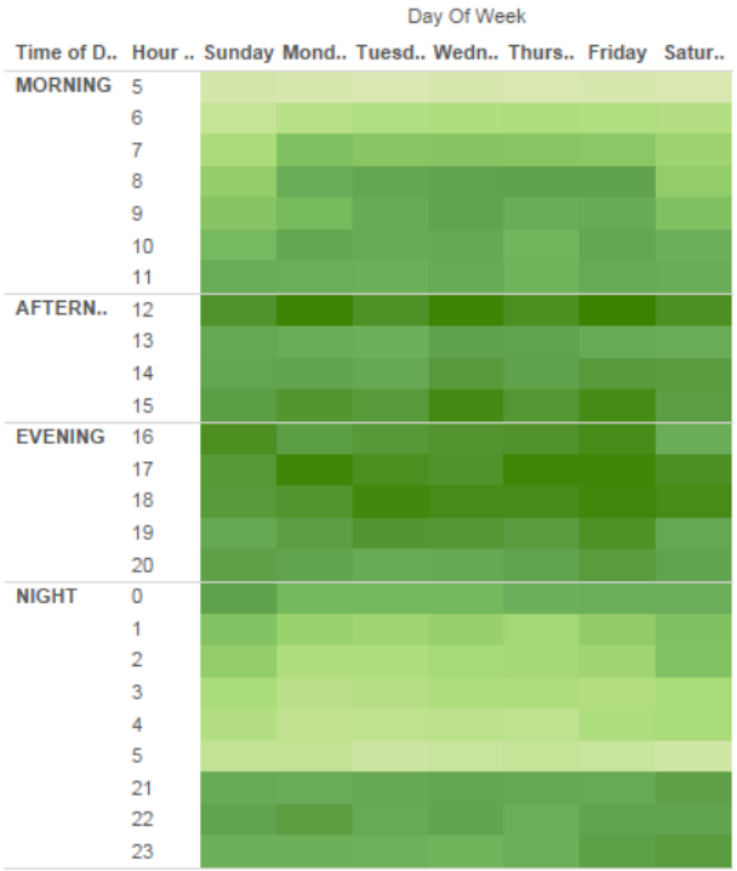
	Pd District / Zip Code	
	PARK	
	94114	94117
Avg. Est Pop 2013	31,397	41,568
Avg. Pop Density	22,028	24,633
Avg. Cost Of Living Index	147.9	149.6
Avg. Unemployment Rate	6.0	6.0
Avg. Land Area	1.4	1.7
Avg. Females	12,512	19,067
Avg. Males	18,885	22,501
Avg. Sex Offenders	7	15

Offense Type
- Higher Level Offenses
- Lower Level Offenses
- NON-CRIMINAL

#Records per Crime Category- Selected PD District/Zip Code/Day of Week&Time of Day



Time of Day/Week Heat Map-By Time of Day for selected PD District, Zip



Number of Crimes - by Address for selected PD District/Zip Code/Day of Week&Time of Day

Address	
700 Block of STANYAN ST	1,226
2600 Block of GEARY BL	730
1200 Block of PAGE ST	612
300 Block of LAGUNAHONDA BL	589
1700 Block of HAIGHT ST	571
1800 Block of HAIGHT ST	565
HAIGHT ST / STANYAN ST	

tableau

Custom Libraries & Methods

Libraries:

- Pandas
- Numpy
- Scipy
- Patsy
- Sklearn

Functions:

- Data Frames
- *Get Dummies*
- *Train/test split?*
- *Label Encoder*

Additional Methods:

- Train/test data
- Data Frames

Description:

Dataset stored as a **Pandas dataframe**. *Categorical features such as time of day, day of week, district, and month encoded using* ***Pandas' get_dummies*** *function* ***Scikit-learn's train_test_split*** *function will be used to set a specific random seed and to ensure reproducibility across different runs. The category of crime encoded using* ***Scikit-learn's LabelEncoder*** *function.*

Classes & Inheritance

Classes defined in the code:

- Login Menu file: Class MainScreen
- Demographics file: Class demographics
- Random Forest file: Class Predictions
- Crime Plot file: Class Crime_Stats
- Crime_stat: Class Mainscreen

Object Orientation Programming

Functionality Description

Data Struc

- Decision Tree Discussion
- Or Stack/Queue
- Others?

```python
#Loginfinal.py

import pandas as pd
import Demographics as dm

class MainScreen:

    def __init__(self):
         self.Badgeinput=1
         self.Passwordinput=1
         self.dfpopo = pd.read_csv("Police.csv",index_col='Badge')

    def empty(df):
        return df.empty

    def plogin(self,Badgeinput,Passwordinput):
        verified=False
        verqry=self.dfpopo.index[self.dfpopo.index==(Badgeinput)]
        # use the true/false array self.dfpopo.index==(Badgeinput) to
        # index into the dataframe. if this is empty, the badge number
        # doesn't exist
        ver2qry=self.dfpopo[self.dfpopo.index==Badgeinput]['Password'].get_values()
        # if verqry does exist, index into the Password column to retrieve
        # the password

        if (len(verqry.tolist()) == 0 | ver2qry != Passwordinput):
            verified==False
            print("\nPlease enter a valid Badge# and Password\n")
            a.LoginMenu()
        else:
            print("\nVerified User...\n")
            verified = True
            a.getName(Badgeinput)
            a.getPDdist(Badgeinput)
            selection = self.Menu1()

        return verified

    def LoginMenu(self):
        Badgeinput=int(input("Badge#:\n"))
        Passwordinput=int(input("Password#:\n"))
        a.plogin(Badgeinput,Passwordinput)

    def listToStringWithoutBrackets(self,list1):
        return str(list1).replace('[','').replace(']','')
```

```
#random_f.py
import numpy as np
import pandas as pd
import scipy as sp
from patsy import dmatrices, dmatrix
from sklearn.calibration import CalibratedClassifierCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, log_loss
from sklearn.decomposition import IncrementalPCA

class Predictions():
        df_train = 0
        df_test = 0
        def __init__(self):
                self.df_train = pd.read_csv("/Users/rushilgoyal/Desktop/train2.csv", parse_dates=["Dates"])
                self.df_test = pd.read_csv('/Users/rushilgoyal/Desktop/test2.csv', parse_dates=['Dates'])


        def Random(self):

                print "number of crime_incidents", len(self.df_train)
                # Dropping Description and Resolution variables since they are not there in test dataset
                self.df_train.drop(['Descript', 'Resolution'], axis=1, inplace=True)
                print "This is how the training crime data looks "
                print self.df_train.head(3)


                # Vectorising Date variable by extracting fetaures for hours, months and years.
                # Also extracting binary features for district and day_of_week
                hours = pd.get_dummies(self.df_train.Dates.map(lambda x: pd.to_datetime(x).hour), prefix="hour")
                months = pd.get_dummies(self.df_train.Dates.map(lambda x: pd.to_datetime(x).month), prefix="month")
                years = pd.get_dummies(self.df_train.Dates.map(lambda x: pd.to_datetime(x).year), prefix="year")
                district = pd.get_dummies(self.df_train["PdDistrict"])
                day_of_week = pd.get_dummies(self.df_train["DayOfWeek"])


                # Concatenating these vectorized variables to generate training dataset
                X_training = pd.concat([self.df_train, hours, months, years, district, day_of_week], axis=1)
                # Dropping "Address" since its already represented by Latitude and Longitude
                # Dropping "Dates", "DayOfWeek" and "PdDistrict" since they have already been vectorized into additional features
                X_training = X_training.drop(["Dates", "Address", "DayOfWeek", "PdDistrict"],  axis = 1)
                print "The new training dataset looks something like this"
                X_training.head(2)
```

```
# Crime_Stat.py
import pandas as pd
import Demographics as dm
import crime_plot as cs
import random_f as rf


# In[2]:


class MainScreen:

    def __init__(self):
         self.Badgeinput=1
         self.Passwordinput=1
         self.dfpopo = pd.read_csv("Police.csv",index_col='Badge')

    def empty(df):
        return df.empty

    def plogin(self,Badgeinput,Passwordinput):
        verified=False
        verqry=self.dfpopo.index[self.dfpopo.index==(Badgeinput)]
        # use the true/false array self.dfpopo.index==(Badgeinput) to
        # index into the dataframe. if this is empty, the badge number
        # doesn't exist
        ver2qry=self.dfpopo[self.dfpopo.index==Badgeinput]['Password'].get_values()
        # if verqry does exist, index into the Password column to retrieve
        # the password

        if (len(verqry.tolist()) == 0 | ver2qry != Passwordinput):
            verified==False
            print("\nPlease enter a valid Badge# and Password\n")
            a.LoginMenu()
        else:
            print("\nVerified User...\n")
            verified = True
            a.getName(Badgeinput)
            a.getPDdist(Badgeinput)
            selection = self.Menu1()

        return verified
    def LoginMenu(self):
        Badgeinput=int(input("Badge#:\n"))
        Passwordinput=int(input("Password#:\n"))
        a.plogin(Badgeinput,Passwordinput)
```

```python
#Crime_plot.py
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn import cross_validation
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from patsy import dmatrices, dmatrix



class Crime_Stats():
        def __init__(self):
                self.train_df = pd.read_csv("/Users/rushilgoyal/Desktop/train2.csv", parse_dates=["Dates"])


        def visualisation(self):
                get_ipython().magic(u'matplotlib inline')
                self.train_df.Category.value_counts().plot(kind="bar")
        # Parsing the dates
                self.train_df["Hour"] = self.train_df["Dates"].map(lambda x: x.hour)
                #The following three visualisations give us an idea of the number of crimes by Day, District and by Hour
                #1.  PLOTTING THE COUNT OF CRIME BY HOUR
                self.train_df["event"] = 1
                hourly_event = self.train_df[["Hour", "event"]].groupby(["Hour"]).count().reset_index()
                hourly_event.plot(kind="bar")
                plt.show()


                #2.  PLOTTING THE COUNT OF CRIME BY DISTRICT
                dist_count = self.train_df.PdDistrict.value_counts()
                plt.figure()
                dist_count.plot(kind="barh")
                plt.ticklabel_format(style='plain', axis='x', scilimits=(0, 0))
                plt.tight_layout()
                plt.show()


                #3.  PLOTTING THE COUNT OF DAY OF WEEK
                day_count = self.train_df.DayOfWeek.value_counts()
                plt.figure()
                day_count.plot(kind="barh")
                plt.ticklabel_format(style='plain', axis='x', scilimits=(0, 0))
                plt.tight_layout()
                plt.show()
```

Data Structures

- Decision Tree Discussion?
- Or Stack/Queue
- Others?

Add small code screen shot to zoom

Description here

File I/O & Iteration

File Name	Input (Contents of file)	Output (Purpose of file)
Police.csv	Police Badge Number, Password, First Name, Last Name, PD District and Alias.	Functions in login verification of officers by comparing badge number and password to associated officer name.
Demographics.csv	PD District, Zip Code, Cost of Living, Estimated Population Density, Land Area, Number of Sex offenders, Unemployment rate, and Total Incidents.	Determine if unemployment and population density impact occurrence of criminal events.
Train.csv	Date, Time, Category of Crime, Description, Day of Week, PD District, Resolution, Address, Latitude and Longitude.	Used to train the system in predicting the category of crime based on real-time data input regarding the officer's district, current date, and time when the officer logs into the system.
Test.csv	Date, Time, Day of Week, PD District, Address, Longitude and Latitude.	Used to test the algorithm generated by training the system.

Iteration

Decision Mal

- Login Authentication (#1)
- Menu Options (#2)

File I/O

```
class MainScreen:

    def __init__(self):
        self.Badgeinput=1
        self.Passwordinput=1
        self.dfpopo = pd.read_csv("Police.csv",index_col='Badge')
```

Login Verification reads "Police" file

```
class Demographics():

    def __init__(self):
        self.dfdemodata=pd.read_csv("Demographics.csv")
```

Demographics class reads "Demographics" file

```
            pf=raw_input("Print to file? Y/N:  ")

            if pf=="y" or pf=="Y":
                f = open('PdDistrictDemographics.csv','w')
                print >>f, ts
                f.close()
                return self.dfdemodata
            else:
                return self.dfdemodata
```

Option to Save the Demographics to file

Decision Making

- Login Authentication (#1)
- Menu Options (#2)

#1

#2

"IF" Statements used for login authetication

```
def plogin(self,Badgeinput,Passwordinput):
    verified=False
    verqry=self.dfpopo.index[self.dfpopo.index==(Badgeinput)]
    # use the true/false array self.dfpopo.index==(Badgeinput) to
    # index into the dataframe. if this is empty, the badge number
    # doesn't exist
    ver2qry=self.dfpopo[self.dfpopo.index==Badgeinput]['Password'].get_values()
    # if verqry does exist, index into the Password column to retrieve
    # the password

    if (len(verqry.tolist()) == 0 | ver2qry != Passwordinput):
        verified==False
        print("\nPlease enter a valid Badge# and Password\n")
        a.LoginMenu()
    else:
        print("\nVerified User...\n")
        verified = True
        a.getName(Badgeinput)
        a.getPDdist(Badgeinput)
        selection = self.Menu1()

    return verified
```

If the user badge and password DO NOT match a record in the Police_Officers.csv file the user is asked to re-enter badge# and password

If the user badge and password match a record in the Police_Officers.csv file the user is "Verified"

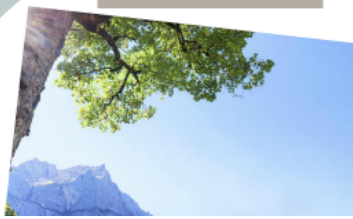
"IF" Statements used for Menu Options

```
def Menu1(self):

    print "\nMenu\n******\n"
    print "1. Demographics\n"
    print "2. Crime Stats\n"
    print "3. Crime Predictions\n"
    print "4. Exit\n"
    menu1input=int(input("Please enter Menu# (1,2,3,4):\n"))

    if menu1input==1:
        b.PdDemo()
        a.Menu1()
```

If the user selects menu option 1 - the "PdDemo" class is called, run, and returns the user back to Menu Options

Testing & Validation

- For our analysis, we will split the dataset into three parts for training, validation and testing with proportion 60%, 20% and 20% respectively.
- Add test results for various models here



screen shot

validation charts

User Experience Design

- Officer logs into system by providing **badge number** and **password**
- After providing the correct credentials, the officer is brought to a **main menu** where he/she can view different types of data.
- Based on the officer's PD district and the current date and time of login, the officer can view the following **crime data**:
  - Crime predictions
  - Crime statistics
  - Demographic-related information
- The officer can choose to **print** the file and/or **save it locally**
- Based on the information provided by the system, decisions regarding locations of **police surveillance** can be made (i.e. increase patrolling in areas that statistically have more crime on that date, time, and location.)

Interface

- Login Screen
- Main Menu
- Submenu

me on

Interface

- Login Screen
- Main Menu
- Submenus?

Add small code screen shot to zoom

Description here

Preventing crime in San Francisco with Predictive Analytics!

Having this data readily at their fingertips will empower officers to make more **efficient decisions** regarding police surveillance, thus **preventing crime** and **reducing costs** for the SFPD.

Preventing crime in San Francisco with Predictive Analytics!

Having this data readily at their fingertips will empower officers to make more **efficient decisions** regarding police surveillance, thus **preventing crime** and **reducing costs** for the SFPD.

About Our Product

Data Source:

Attributes:

Classes & objects:

Libraries:

Methods:

Predictive Models:

Visualizations:

Sample Code:

San Francisco Crime Prevention

Nicole Dziedic, Siddhant Garg,
Rushil Goyal, Danielle Henriquez