Data-Driven Hotspot Identification

By Group -09
Vidhi Shiyani
Komal Dodiya
Aniket Patel

NEW YORK STATE | Division of Criminal Justice Services

DATA SOURCE

https://opendata.cityofnewyork.us/
New York State Division of Criminal Justice Services (DCJS)

PROBLEM STATEMENT

To classify the hotspots in the New York State county where the number of criminals aged 18 or older is significantly higher, so that the government can implement measures to reduce the conversion rate of juvenile offenders.

We often hear about rising crime rates, but one of the biggest concerns is the transition of juvenile offenders into adult criminals. If we can identify where adult crimes are concentrated, we can better understand the environment influencing youth and take preventive action.

Why Dataset matters the most ?

1) Juvenile crime can turn into adult crime if early intervention is not applied.
2) Identifying High risk counties allows the government to focus resources.
3) Helps in designing Youth Programs, community support and policy strategies.
4) Data Driven Decision making makes crime inventions smarter and more effective.

ABOUT THE DATASET

Adult Arrests (18 and Older) by County – Beginning 1970

- Total Records: 3,410
- Time Period Covered: 1970 – 2024 (55 years of data)
- Geographic Coverage: 62 counties across New York State
- Population Group: Adults aged 18 years and older
- Observation Unit: County-Year combination (e.g., Albany-1970)

EXPLORING THE DATA

It shows sample values for Albany County from 1970–1974, helping to understand how arrest counts are distributed across different categories (felonies and misdemeanors).

```
1 data.head()
```

	County	Year	Total	Felony Total	Drug Felony	Violent Felony	DWI Felony	Other Felony	Misdemeanor Total	Drug Misdemeanor	DWI Misdemeanor	Property Misdemeanor	Other Misdemeanor
0	Albany	1970	1,032	563	90	155	4	314	469	179	47	73	170
1	Albany	1971	1,566	693	119	199	6	369	873	179	111	210	373
2	Albany	1972	2,604	864	181	215	8	460	1,740	251	295	417	777
3	Albany	1973	3,055	955	213	230	28	484	2,100	298	491	506	805
4	Albany	1974	3,587	1,070	225	244	17	584	2,517	354	610	670	883

```
data.shape
```

```
(3410, 13)
```

The dataset contains 3,410 rows and 13 columns.
Each row represents arrest data for a specific county and year, while the columns represent different offense categories and totals.

EXPLORING THE DATA

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3410 entries, 0 to 3409
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   County                 3410 non-null   object
 1   Year                   3410 non-null   int64 
 2   Total                  3410 non-null   object
 3   Felony Total           3410 non-null   object
 4   Drug Felony            3410 non-null   object
 5   Violent Felony         3410 non-null   object
 6   DWI Felony             3410 non-null   int64 
 7   Other Felony           3410 non-null   object
 8   Misdemeanor Total      3410 non-null   object
 9   Drug Misdemeanor       3410 non-null   object
 10  DWI Misdemeanor        3410 non-null   object
 11  Property Misdemeanor   3410 non-null   object
 12  Other Misdemeanor      3410 non-null   object
dtypes: int64(2), object(11)
memory usage: 346.5+ KB
```

The summary shows the data types, non-null counts, and memory usage.
It reveals that there are no missing values, but most columns are stored as objects (strings) instead of numeric types—indicating the need for data cleaning before analysis.

EXPLORING THE DATA

This table gives descriptive statistics of dataset. Total records of 3410, The average number of DWI felony cases is around 67, but the maximum goes up to 613, showing a very high variation across counties.
This tells us that some counties are extremely high crime zones compared to others, which is why hotspot classification is necessary.

```
1 data.describe()
```

--NORMAL--

	Year	DWI Felony
count	3410.000000	3410.000000
mean	1997.000000	66.691202
std	15.876836	88.229111
min	1970.000000	0.000000
25%	1983.000000	17.000000
50%	1997.000000	37.000000
75%	2011.000000	76.000000
max	2024.000000	613.000000

FEATURE ENGINEERING

```
#All the arrest numbers are in numeric form but are falling in object data type cause of ','
cols_to_convert = ['Total', 'Felony Total', 'Violent Felony', 'Drug Felony', 'Other Felony',
                   'Misdemeanor Total', 'Drug Misdemeanor', 'DWI Misdemeanor',
                   'Property Misdemeanor', 'Other Misdemeanor']

#Converting columns to numeric
for col in cols_to_convert:
    data[col] = pd.to_numeric(data[col].str.replace(',', ''), errors='coerce')
```

We converted the features consisting of object data type into numeric data type by using the string replacement function.

	County	Year	Total	Felony Total	Drug Felony	Violent Felony	DWI Felony	Other Felony	Misdemeanor Total	Drug Misdemeanor	DWI Misdemeanor	Property Misdemeanor	Other Misdemeanor	Total_Arrests_Lag1
0	Albany	1970	1032	563	90	155	4	314	469	179	47	73	170	NaN
1	Albany	1971	1566	693	119	199	6	369	873	179	111	210	373	1032.0
2	Albany	1972	2604	864	181	215	8	460	1740	251	295	417	777	1566.0
3	Albany	1973	3055	955	213	230	28	484	2100	298	491	506	805	2604.0
4	Albany	1974	3587	1070	225	244	17	584	2517	354	610	670	883	3055.0

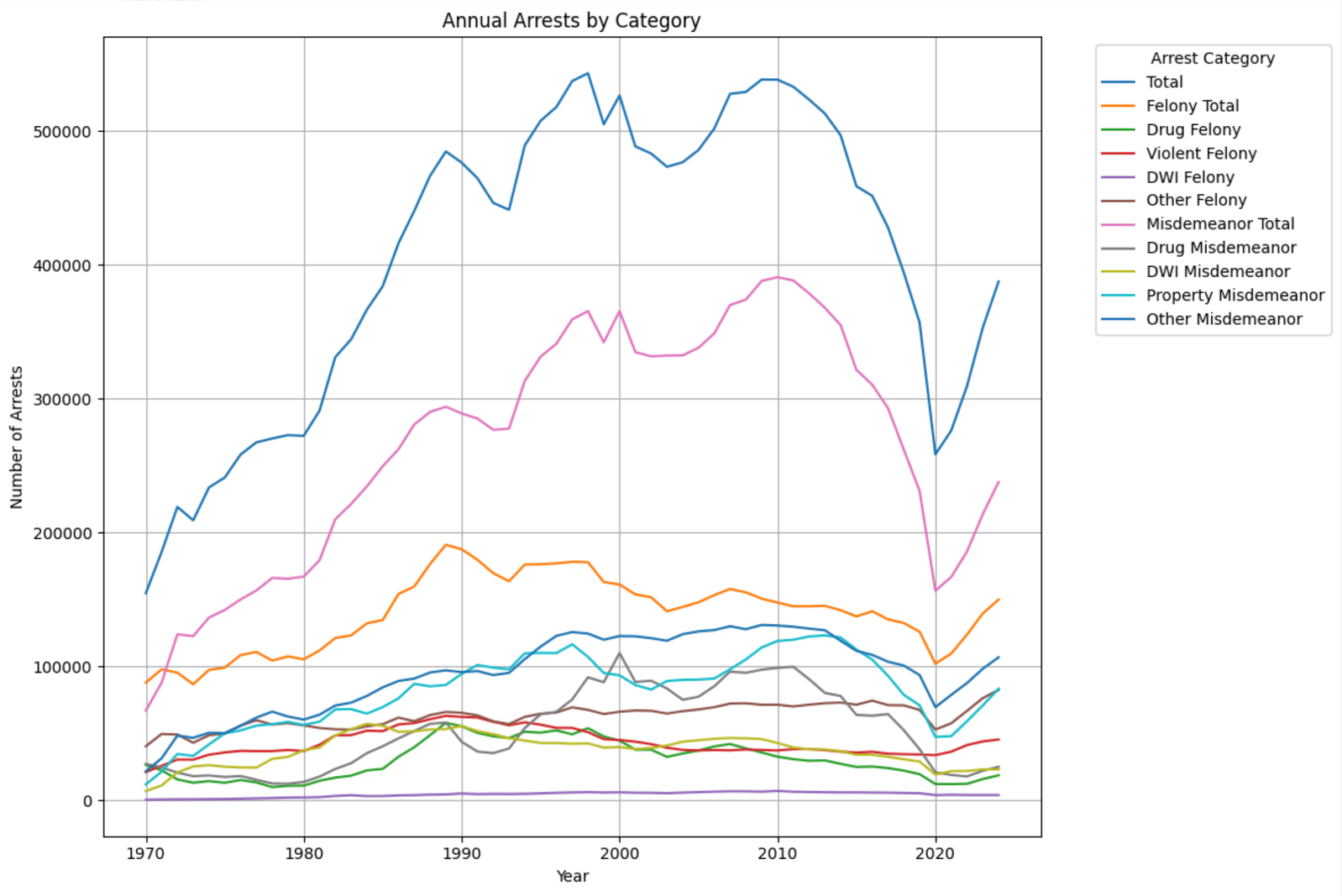
Further we added a dummy column titled as ‘Total_Arrests_Lag1’, where the data of previous year’s Total is added for better accuracy of predictive models like cumulative frequency.

FEATURE ENGINEERING

DWI Felony	Other Felony	Misdemeanor Total	Drug Misdemeanor	DWI Misdemeanor	...	County_Sullivan	County_Tioga	County_Tompkins	County_Ulster	County_Warren	County_Washington	County_Wayne
4	314	469	179	47	...	False	False	False	False	False	False	False
6	369	873	179	111	...	False	False	False	False	False	False	False
8	460	1740	251	295	...	False	False	False	False	False	False	False
28	484	2100	298	491	...	False	False	False	False	False	False	False
17	584	2517	354	610	...	False	False	False	False	False	False	False

We then used One-Hot Encoding method to create dummy features for all the counties and the dummy features we created are in boolean , basically it creates new Binary columns for each category.

INITIAL VISUALIZATION



We grouped the data for all types of crime and total of all the years from 1970 - 2024 almost 54 years, to visualize it in a line graph for easier and better understanding.

HANDLING MISSING VALUES

	0
Year	0
Total	0
Felony Total	0
Drug Felony	0
Violent Felony	0
...	...
County_Washington	0
County_Wayne	0
County_Westchester	0
County_Wyoming	0
County_Yates	0

74 rows × 1 columns

We then handled missing values in the data. We didn't have any missing values as such but once we added the 'Total_Arrests_Lag' feature the value of it for the year 1970 became NaN for all counties which we replaced with '0' using 'fillna' feature.

```
train_data.fillna(0, inplace=True)

test_data.fillna(0, inplace=True)

print("Training data shape after handling missing values:", train_data.shape)
print("Testing data shape after handling missing values:", test_data.shape)
```

```
Training data shape after handling missing values: (3100, 74)
Testing data shape after handling missing values: (310, 74)
```

SPLITING DATA INTO TRAINING AND TESTING

```
Training data shape: (3100, 74)
Testing data shape: (310, 74)
```

Now, using the ‘train_test_split’ feature we split the data into training and testing datasets where data from 2020 to 2024 is taken in consideration for test case.

```
1 split_year = 2020
2 train_data = grouped_data_encoded[grouped_data_encoded['Year'] < split_year].copy()
3 test_data = grouped_data_encoded[grouped_data_encoded['Year'] >= split_year].copy()
4
5 print("Training data shape:", train_data.shape)
6 print("Testing data shape:", test_data.shape)
```

```
Training data shape: (3100, 74)
Testing data shape: (310, 74)
```

LINEAR REGRESSION

To forecast future total adult arrests in each county based on historical arrest patterns and category-level arrest data.

```
1 X_train = train_data.drop('Total', axis=1)
2 y_train = train_data['Total']
3
4 X_test = test_data.drop('Total', axis=1)
5 y_test = test_data['Total']
6
7 print("X_train shape:", X_train.shape)
8 print("y_train shape:", y_train.shape)
9 print("X_test shape:", X_test.shape)
10 print("y_test shape:", y_test.shape)
--NORMAL--
```

```
X_train shape: (3100, 73)
y_train shape: (3100,)
X_test shape: (310, 73)
y_test shape: (310,)
```

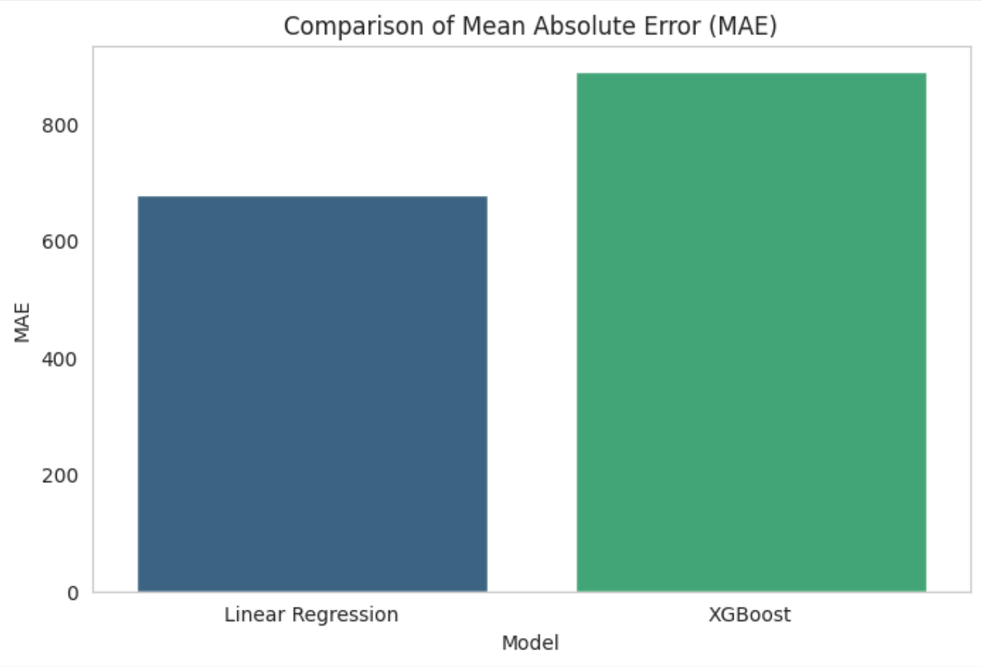
But for regression, I separated the output column (‘Total’ arrests) from the input features.

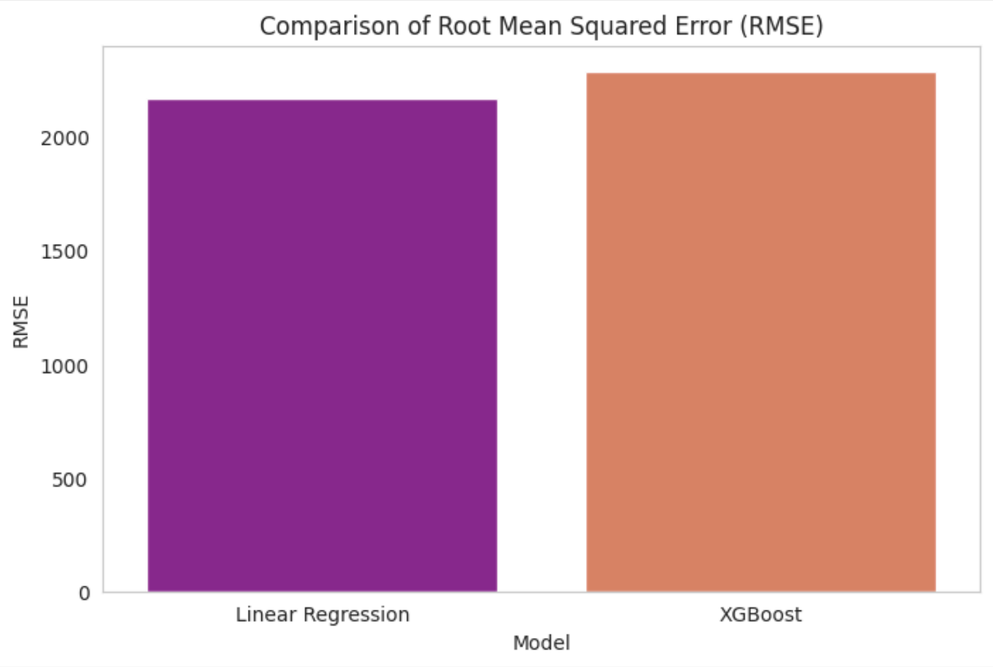
Comparing both Linear Regression and XG Boost Model

Model Performance Comparison:

	Model	MAE	RMSE
0	Linear Regression	678.415347	2166.423733
1	XGBoost	888.271362	2287.730207

Compared to the mean of 5109 we were able to achieve an average error of 678 which is roughly 13% of the mean and with XGBoost its roughly 17.38% of the mean. Compared to the large range of values an error of 678 and 888 is relatively small.





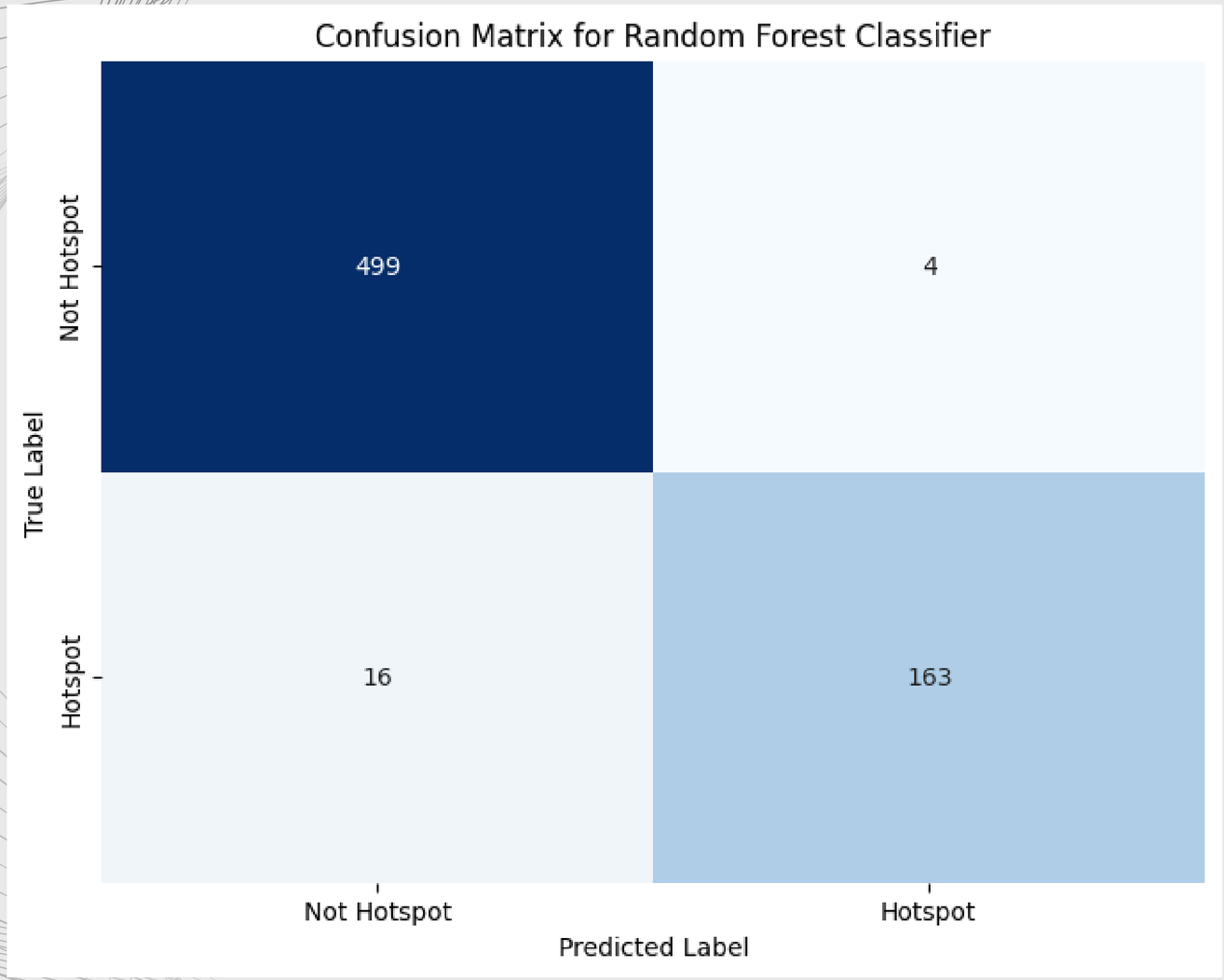
RANDOM FOREST CLASSIFIER

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split

hotspot_threshold = data['Total'].quantile(0.75)
data['Hotspot'] = (data['Total'] > hotspot_threshold).astype(int)

classification_features = data.drop(['Total', 'Hotspot', 'Felony Total', 'Drug Felony', 'Violent Felony', 'DWI Felony',
                                     'Other Felony', 'Misdemeanor Total', 'Drug Misdemeanor',
                                     'DWI Misdemeanor', 'Property Misdemeanor', 'Other Misdemeanor'], axis=1)

classification_features_encoded = pd.get_dummies(classification_features, columns=['County'], drop_first=True)

classification_target = data['Hotspot']
```

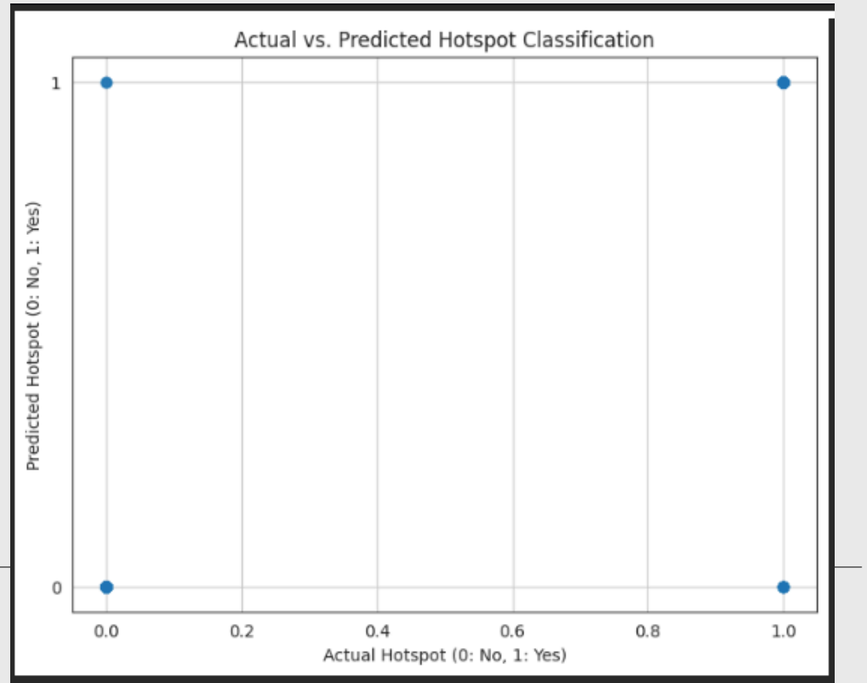
```
Random Forest Classifier Accuracy: 0.97

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       503
           1       0.98      0.91      0.94       179

    accuracy                           0.97       682
   macro avg       0.97      0.95      0.96       682
weighted avg       0.97      0.97      0.97       682
```

We used Random Forest Classifier model as a complex model for classification and were able to achieve an **accuracy of 0.97 along with recall of 0.99.**

RANDOM FOREST CLASSIFIER



We can clearly see the efficiency of the random forest model through the confusion matrix visualization. Only 20 samples were misidentified from which we can state that the model is highly efficient for our dataset and problem statement.



Counties Ranked by Proportion of Years as Hotspots

```
hotspot_counts = data.groupby('County')['Hotspot'].mean()
ranked_counties = hotspot_counts.sort_values(ascending=False)

print("Counties Ranked by Proportion of Years as Hotspots:")
display(ranked_counties)
```

```
Counties Ranked by Proportion of Years as Hotspots:
```

County	Hotspot
Bronx	1.0
Erie	1.0
Kings	1.0
Nassau	1.0
Queens	1.0
...	...
Tompkins	0.0
Washington	0.0
Wayne	0.0
Wyoming	0.0
Yates	0.0

62 rows × 1 columns

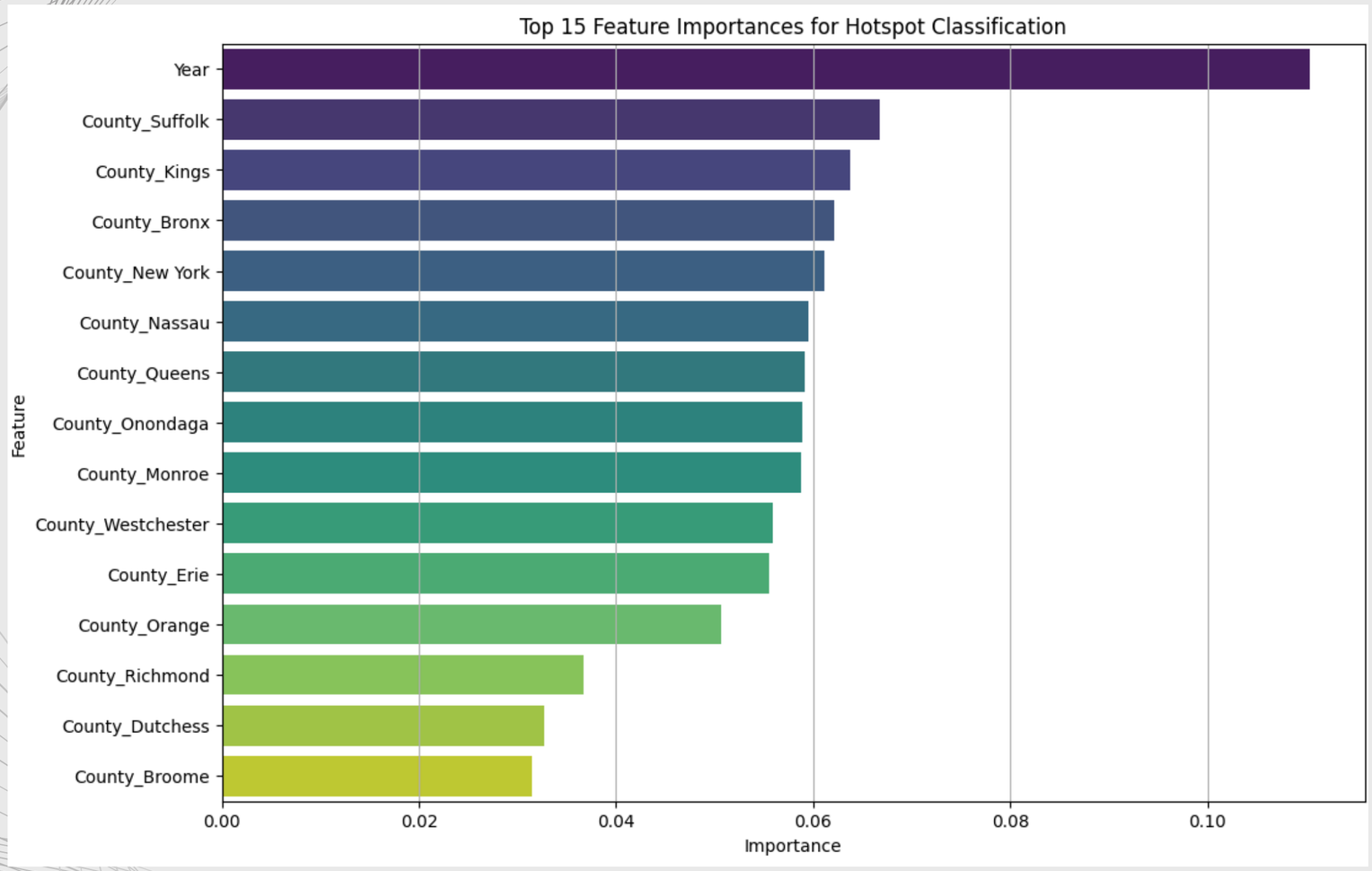
This table displays New York counties ranked based on how often they were identified as crime hotspots over the years.

“Hotspot value 1.0” means the county was a hotspot every year in the dataset (100% of the time).

“Hotspot value 0.0” means the county was never a hotspot in any year.

This helps us clearly identify regions that are chronically high-risk vs consistently safe, giving policymakers a priority list for interventions.

FEATURE IMPORTANCE OF RFC



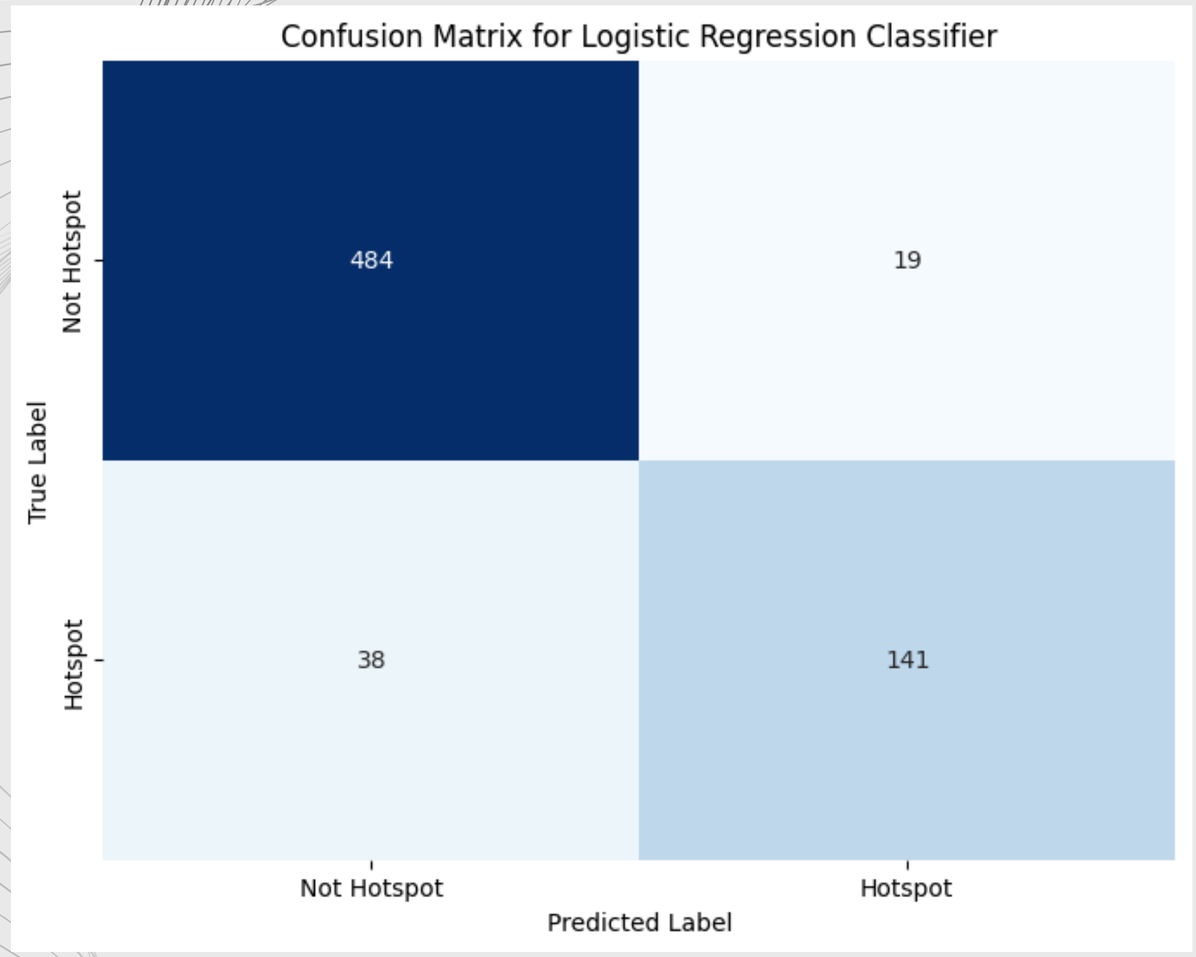
Here’s the feature importance graph for the top 15 features as per the random forest classifier.

LOGISTIC REGRESSION

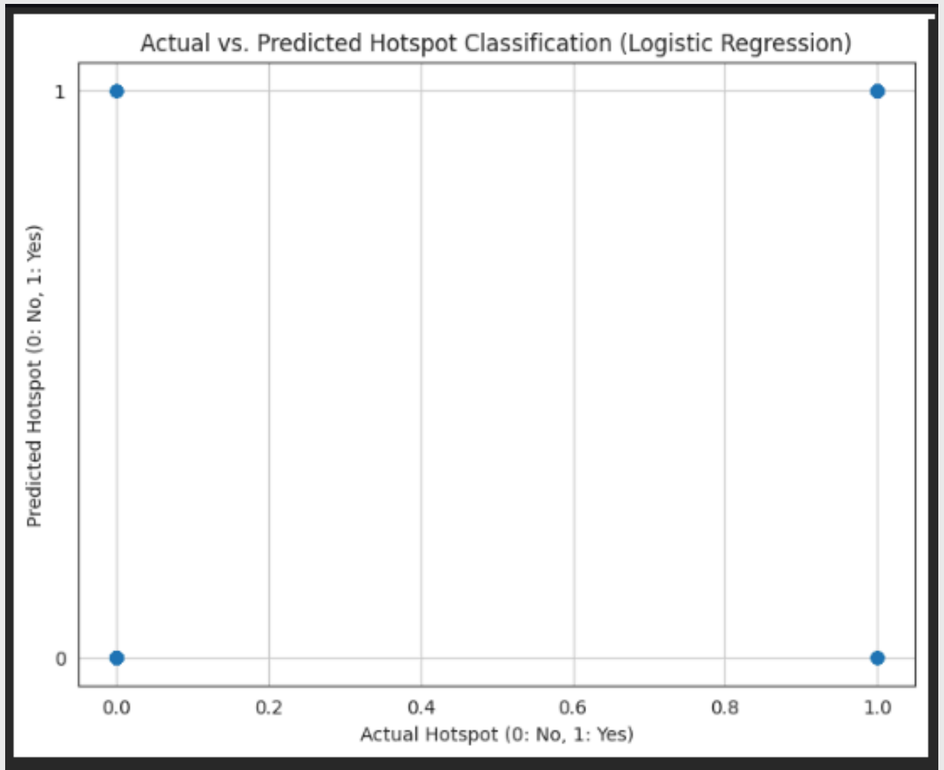
```
Logistic Regression Classifier Accuracy: 0.92

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.96      0.94       503
           1       0.88      0.79      0.83       179

    accuracy                           0.92       682
   macro avg       0.90      0.87      0.89       682
weighted avg       0.92      0.92      0.91       682
```

We used Logistic Regression Classifier model as a simpler model for classification and were able to achieve an accuracy of 0.93 along with recall of 0.96.

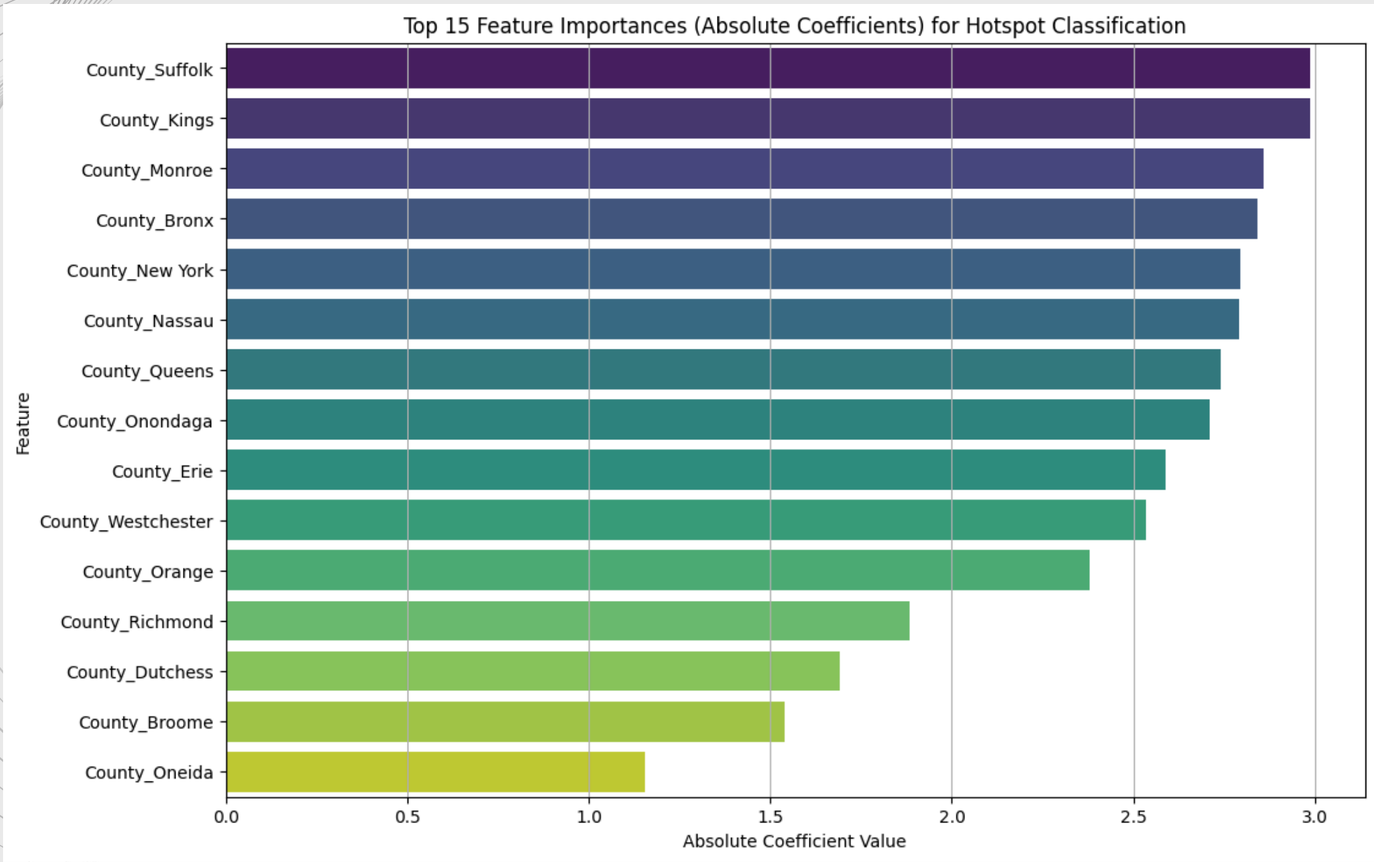
LOGISTIC REGRESSION



We can clearly see the efficiency of the logistic regression model through the confusion matrix visualization. Only 50 samples were misidentified from which we can state that the model is efficient for our dataset and problem statement but not as efficient as the Random Forest Classifier.



FEATURE IMPORTANCE OF LR

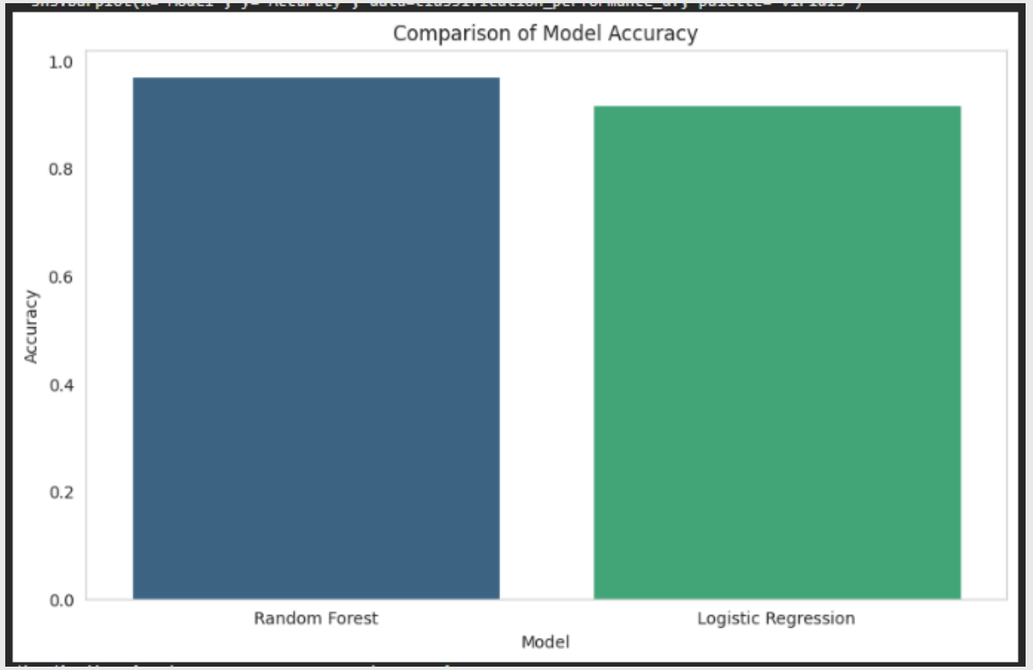


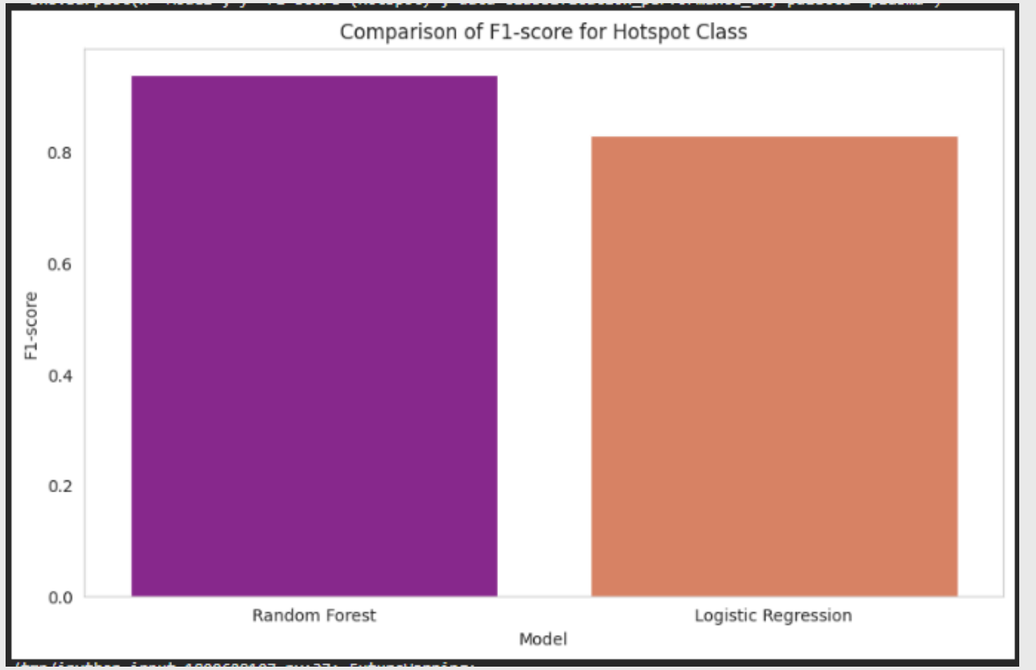
Here’s the feature importance graph for the top 15 features as per the logistic regression model.

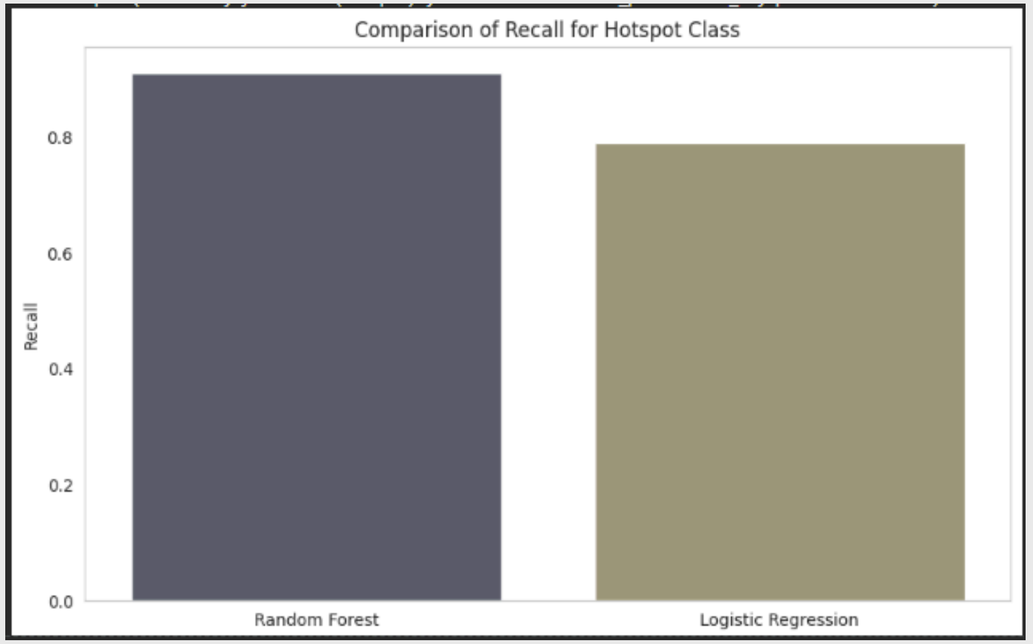
MODEL COMPARISON

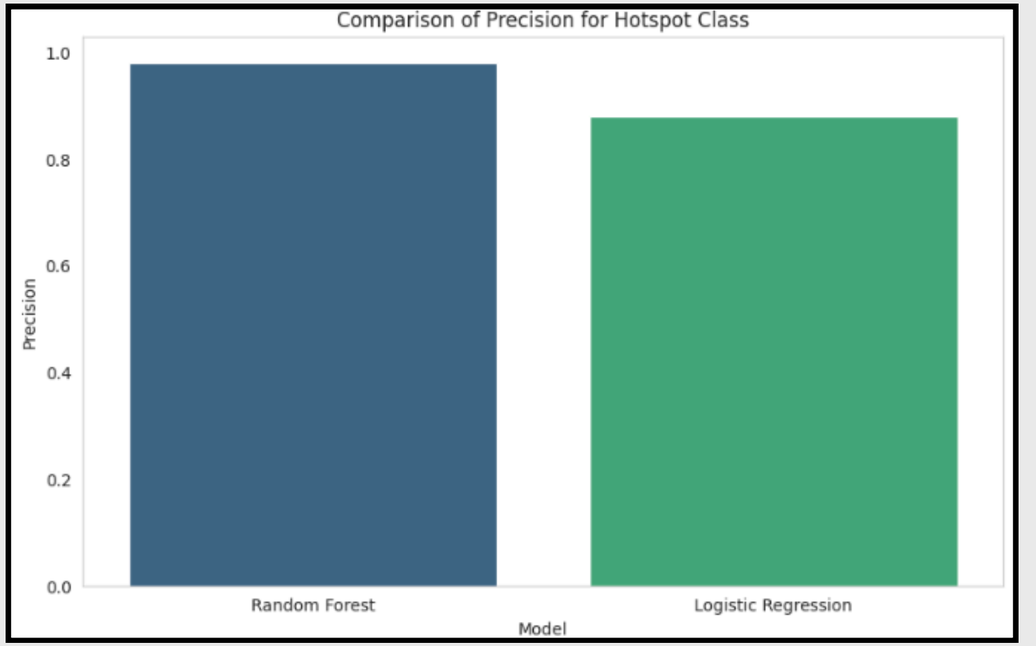
	Model	Accuracy	Precision (Hotspot)	Recall (Hotspot)	F1-score (Hotspot)
0	Random Forest	0.970674	0.98	0.91	0.94
1	Logistic Regression	0.916422	0.88	0.79	0.83

Here’s the comparison of both the models that we used to identify the hotspot counties of New York state based on past records. We are clearly able to make out that Random Forest classifier works better for our dataset and problem statement.

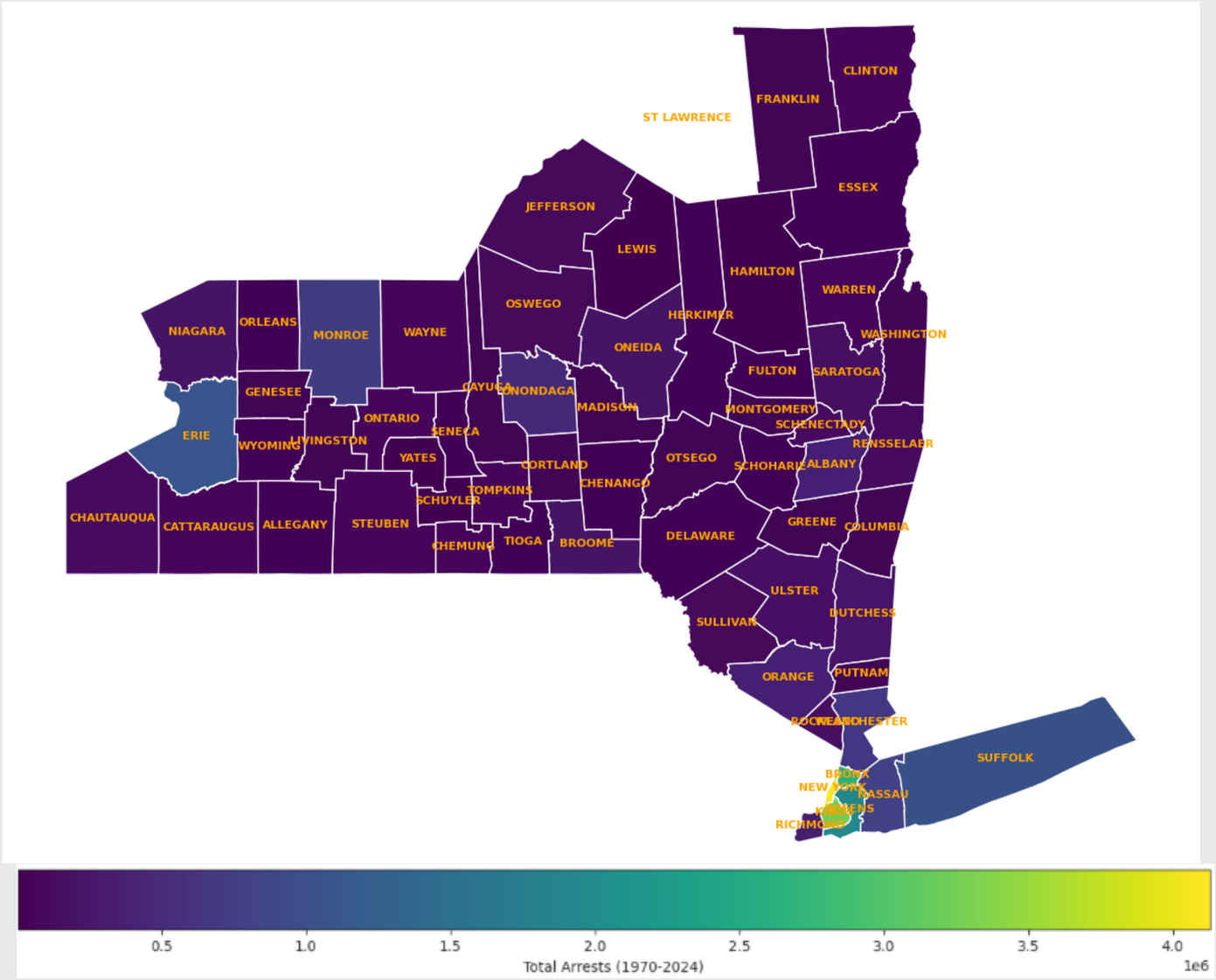

Comparison of Model Accuracy
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
Random Forest
Logistic Regression
Model


Comparison of F1-score for Hotspot Class
F1-score
0.8
0.6
0.4
0.2
0.0
Random Forest
Logistic Regression
Model


Comparison of Recall for Hotspot Class
Recall
0.8
0.6
0.4
0.2
0.0
Random Forest
Logistic Regression

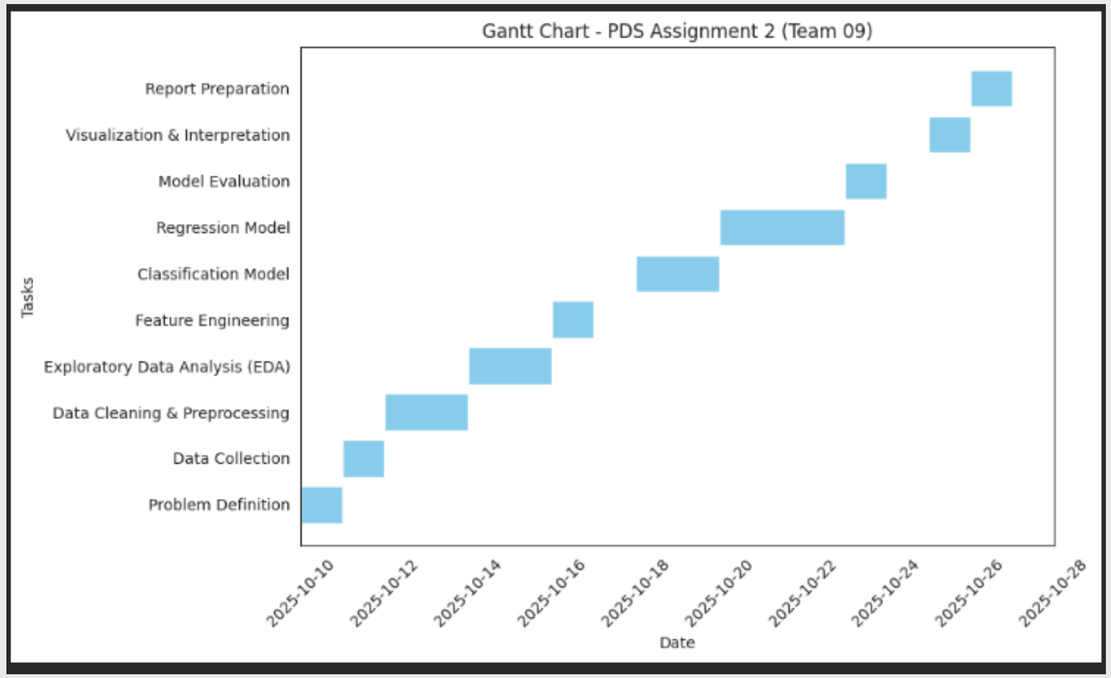

Comparison of Precision for Hotspot Class
Precision
1.0
0.8
0.6
0.4
0.2
0.0
Random Forest
Logistic Regression
Model

VISUALIZATION OF HOTSPOTS



TEAM - 9 : GANTT CHART STRUCTURE

Task	Description	Start Date	End Date	Duration	Responsible
1. Problem Definition	Define classification & regression objectives	Oct 10	Oct 11	1 day	Komal
2. Data Collection	Import dataset (Adult Arrests CSV)	Oct 11	Oct 12	1 day	Vidhi
3. Data Cleaning & Preprocessing	Handle missing values, filter columns, handle data types	Oct 12	Oct 14	3 days	Aniket
4. Exploratory Data Analysis (EDA)	Visualize trends, distributions, correlations	Oct 14	Oct 16	3 days	Komal
5. Feature Engineering	Select key features for both models	Oct 16	Oct 17	2 days	Vidhi
6. Classification Model	Train and evaluate classification model (e.g., Logistic Regression, Random Forest)	Oct 18	Oct 20	3 days	Aniket
7. Regression Model	Build time series or regression model for arrests prediction	Oct 20	Oct 23	4 days	Komal
8. Model Evaluation	Compare models using accuracy, RMSE, R^2, etc.	Oct 23	Oct 24	2 days	Team
9. Visualization & Interpretation	Display hotspot map, prediction graphs	Oct 25	Oct 26	2 days	Vidhi
10. Report Preparation	Summarize results, create presentation/report	Oct 26	Oct 27	2 days	Team



THANK YOU !