PPOL 5204: Data Science II
Group Assignment 1

Ibadat Jarg, Emily Zhao, Xiao Xu

Question 1
Communities and Crime

Data Description

- Data is a compilation of:
  1. LEMAS stats (1990)
  2. US Census (1990)
  3. FBI Crime stats (1995)
- The data has over **122 features** that can be used to predict **Violent Crime per capita** (Target Variable) for **1995 counties**

- Contains variables that describe demographic & policing practices for each county
- Numerics have been **normalized to [0,1]**
  - Any values more than 3 standard deviations away from mean have been assigned as 0 or 1

name	state	county	community	communityname	fold	population	householdsize	racepctblack	racePctWhite	racePctAsian	...	LandArea	PopDens
0	8	NaN	NaN	Lakewoodcity	1	0.19	0.33	0.02	0.90	0.12	...	0.12	0.26
1	53	NaN	NaN	Tukwilacity	1	0.00	0.16	0.12	0.74	0.45	...	0.02	0.12
2	24	NaN	NaN	Aberdeentown	1	0.00	0.42	0.49	0.56	0.17	...	0.01	0.21
3	34	5	81440	Willingborotownship	1	0.04	0.77	1.00	0.08	0.12	...	0.02	0.39
4	42	95	6096	Bethlehemtownship	1	0.01	0.55	0.02	0.95	0.09	...	0.04	0.09
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1989	12	NaN	NaN	TempleTerracecity	10	0.01	0.40	0.10	0.87	0.12	...	0.01	0.28
1990	6	NaN	NaN	Seasidecity	10	0.05	0.96	0.46	0.28	0.83	...	0.02	0.37
1991	9	9	80070	Waterburytown	10	0.16	0.37	0.25	0.69	0.04	...	0.08	0.32
1992	25	17	72600	Walthamcity	10	0.08	0.51	0.06	0.87	0.22	...	0.03	0.38
1993	6	NaN	NaN	Ontariocity	10	0.20	0.78	0.14	0.46	0.24	...	0.11	0.30

Preprocessing steps

1. We dropped several variables that were not relevant to our prediction model:

   State, fold, county, community, communityname

2. The following features: had missing values

```
Index(['OtherPerCap', 'LemasSwornFT', 'LemasSwFTPerPop', 'LemasSwFTFieldOps',
       'LemasSwFTFieldPerPop', 'LemasTotalReq', 'LemasTotReqPerPop',
       'PolicReqPerOffic', 'PolicPerPop', 'RacialMatchCommPol',
       'PctPolicWhite', 'PctPolicBlack', 'PctPolicHisp', 'PctPolicAsian',
       'PctPolicMinor', 'OfficAssgnDrugUnits', 'NumKindsDrugsSeiz',
       'PolicAveOTWorked', 'PolicCars', 'PolicOperBudg', 'LemasPctPolicOnPatr',
       'LemasGangUnitDeploy', 'PolicBudgPerPop'],
      dtype='object', name='name')
```

- We chose to use an **Interative imputer** to deal with our missing values
  - Why? Missing values are likely highly interrelated

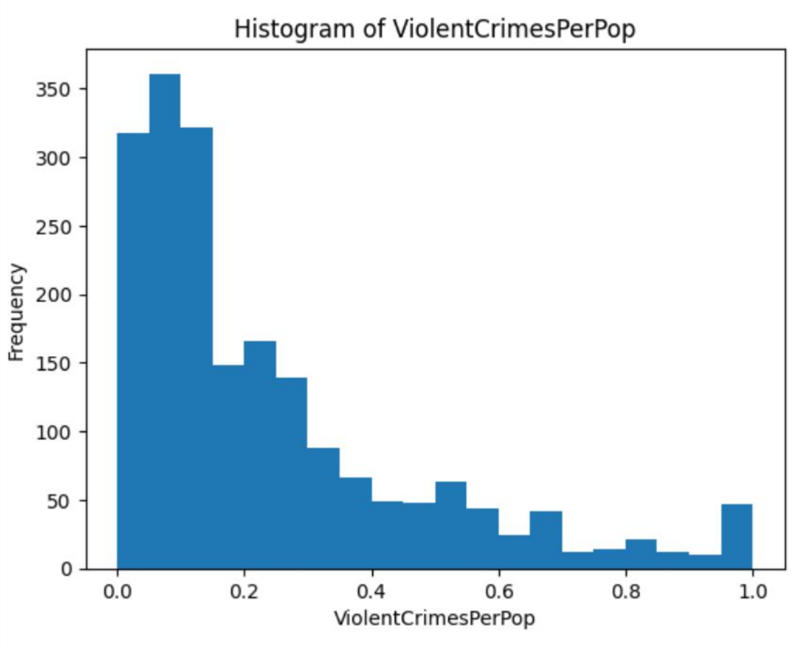
3. Feature engineering was accomplished by:

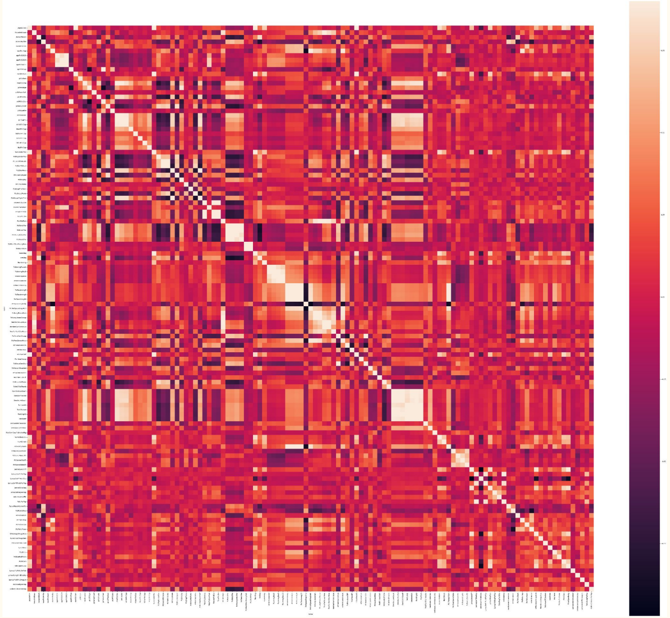
   (i) Correlation heatmap

   (ii) Plots for each feature against target

We created 3 new features: numbUrbansq, Pov_Unemp_Interaction and pctUnderPovsq

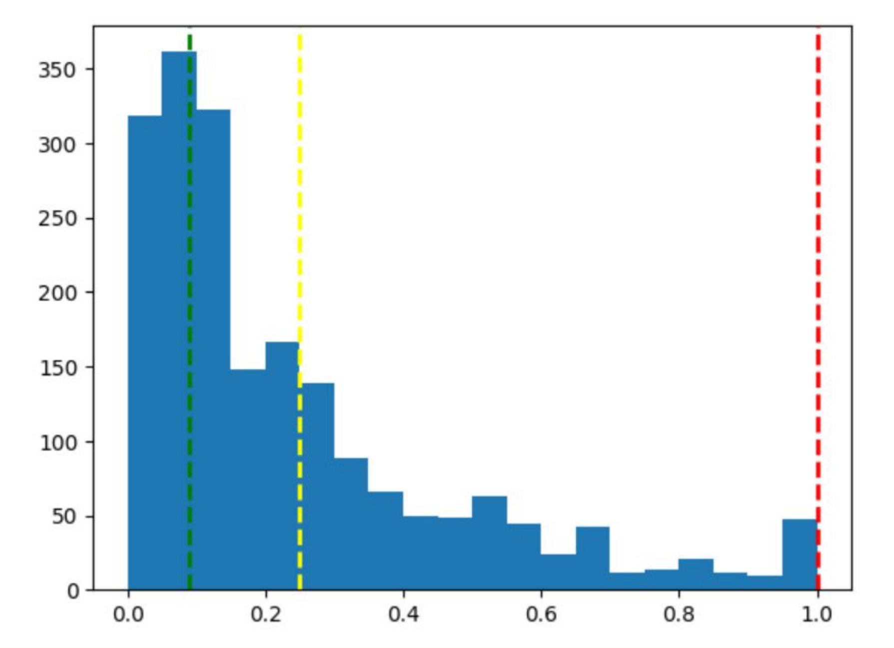
Curated Data





Model Implementation

- We opted for a regularized regression model, specifically **Elastic Net**
- We implemented a 70-30 training test split
- For our **discretization** strategy we chose to categorize violent crime in **Low, Medium and High** based on 0.33 and 0.66 quantiles
- The 2 classifiers we used were a Random Forest Classifier and Logistic regression



Model Evaluation

To evaluate our regression model we used the **RMSE** and **R^2** see which of our 3 models were the most suitable.

Ultimately after testing a wide range or alphas and L1 ratios **Elastic net** bore the best results with **RMSE = 0.1179** and **R^2 = 0.7093**

```
LASSO best alpha: {'alpha': 0.0001}
LASSO RMSE: 0.11810897044084243
LASSO R^2: 0.7086508323598422
```

```
Ridge best alpha: {'alpha': 1}
Ridge RMSE: 0.11891637544746604
Ridge R^2: 0.7046538315040347
```

```
Elastic Net best params: {'alpha': 0.0001, 'l1_ratio': 0.9}
Elastic Net RMSE: 0.11796529361838103
Elastic Net R^2: 0.709359240223244
```

To evaluate our Classification model we used **precision, recall** and **F1-scores**

Overall we preferred the **Multinomial Logistic** regression higher metrics across the board compared to random forests

```
=== Random Forest ===
[[165  38   1]
 [ 56  95  48]
 [  5  38 153]]
              precision    recall  f1-score   support

           0       0.73      0.81      0.77       204
           1       0.56      0.48      0.51       199
           2       0.76      0.78      0.77       196

    accuracy                           0.69       599
   macro avg       0.68      0.69      0.68       599
weighted avg       0.68      0.69      0.68       599
```

```
=== Multinomial Logistic Regression ===
[[169  34   1]
 [ 58  97  44]
 [  5  35 156]]
              precision    recall  f1-score   support

           0       0.73      0.83      0.78       204
           1       0.58      0.49      0.53       199
           2       0.78      0.80      0.79       196

    accuracy                           0.70       599
   macro avg       0.70      0.70      0.70       599
weighted avg       0.70      0.70      0.70       599
```

Bias, Ethical Considerations and further research

Methodological Evaluation: Many demographic variables overlap making model latch onto redundant noise, Missing data and mismatches in years , Overall after tunning model was satisfactory

Future Research: More features related to mental health, drug abuse and gun ownership

Ethics & Model Deployment: Our model should NOT be used for invasive and harmful policy measures such as increased policing. Recommend information not invasive interventions eg. rehab services

Bias: Reporting Bias can be problematic, factors such as reduced police trust or systematic over or under policing leads to measurement error

Question 2
Malware Detection

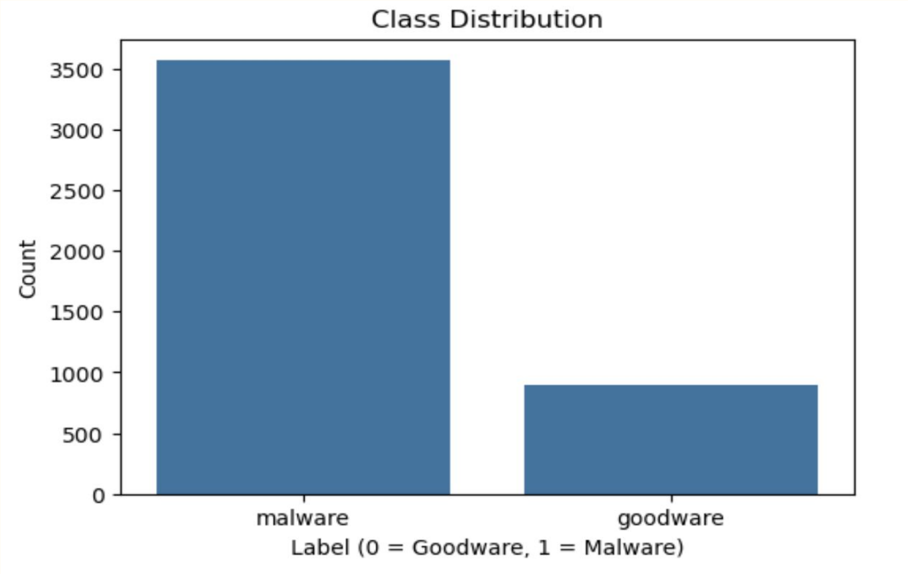
Data Description

The dataset is a collection of API calls, permissions or behaviours of **241 features** for **4465 samples**.

The goal is to **classify** these applications are malware or goodware (Labels). Classified as Label = 1 as Malware and Label = 0 as goodware.

The data is already binarized, hence we do not need to engage in that step of preprocessing

We do see that the labels are heavily imbalanced approximately in 8:2 ratio



Preprocessing and Curation steps

We will split our data in **70-15-15** training, validation & testing set

To deal with our imbalanced dataset we will implement **SMOTE** to increase the number of sample for the underrepresented class

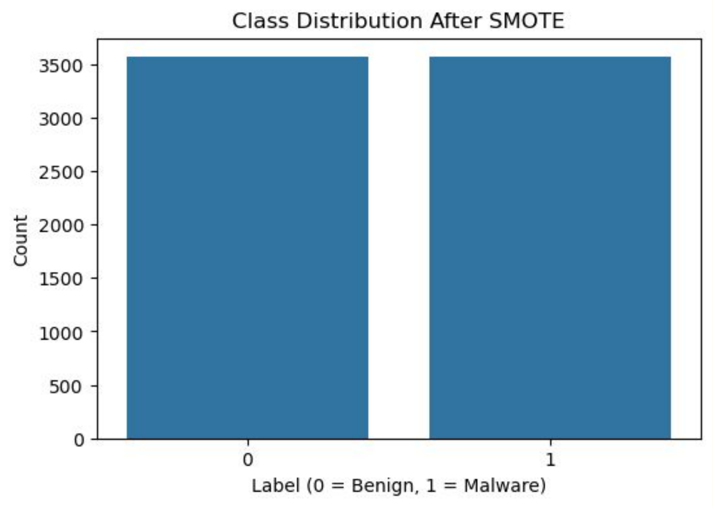
- Note: Only applied to training set

Use two-step split:

1. Split original data into 70% training, 30% temp set
2. Split temp set equally into validation and test sets

- Stratified sampling used to maintain class balance across all subsets

- **Verification:** Checked class distributions and subset sizes for balance



Classification Strategy

We used **Random Forest** and **XGBoost** classifiers to detect malware.

Both models performed exceptionally well with:

- Accuracy: ~99.1%
- F1-scores > 0.97 across both classes
- AUC-ROC > 0.998, indicating excellent class separation

Confusion matrices show very few misclassifications:

- Random Forest:
    a. False Positives (Benign -> Malware): 2
    b. False Negatives (Malware -> Benign): 4
- XGBoost:
    a. Same number of misclassifications as RF
- Confusion matrices indicate minimal misclassification and robust model reliability.

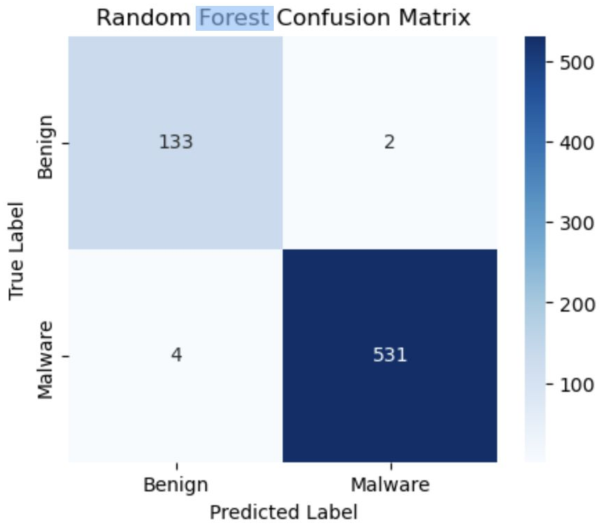
Overall, **Random Forest** offers similar predictive power with better interpretability and lower computational cost.

Model Evaluation

🔷 Random Forest Classification Report:

	precision	recall	f1-score	support
0.0	0.9708	0.9852	0.9779	135
1.0	0.9962	0.9925	0.9944	535
accuracy			0.9910	670
macro avg	0.9835	0.9889	0.9862	670
weighted avg	0.9911	0.9910	0.9911	670

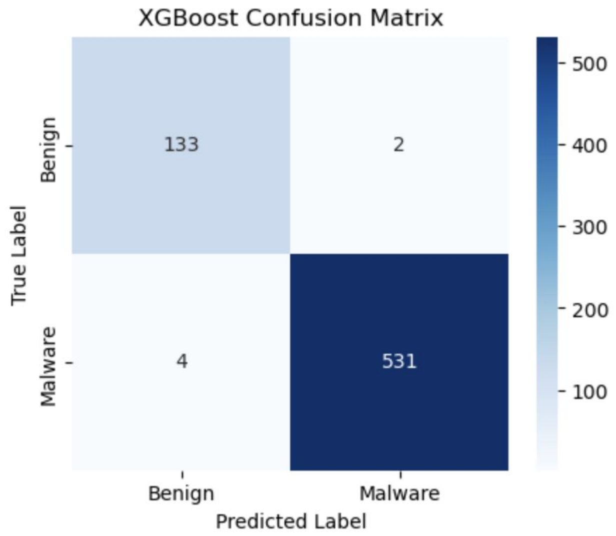
AUC-ROC Score: 0.9989



🔷 XGBoost Classification Report:

	precision	recall	f1-score	support
0.0	0.9708	0.9852	0.9779	135
1.0	0.9962	0.9925	0.9944	535
accuracy			0.9910	670
macro avg	0.9835	0.9889	0.9862	670
weighted avg	0.9911	0.9910	0.9911	670

AUC-ROC Score: 0.9987



Model Optimization & Evaluation

Applied **Grid Search CV** to tune hyperparameters for both models

- **Random Forest:** n_estimators, max_depth, min_samples_split
- **XGBoost:** n_estimators, max_depth, learning_rate

🔸 **Best Random Forest:** n_estimators = 50, min_samples_split = 5

🔸 **Best XGBoost:** n_estimators = 200, max_depth = 6, learning_rate = 0.1

Evaluation focused on **F1-score** to balance **precision** and **recall**

Results:

- **Recall ≈ 0.99** → excellent malware detection
- **Precision ≈ 0.96–1.00** → very few benign misclassified
- **AUC-ROC > 0.998** → strong class separation

Model Interpretability & Real-World Insights

We Extracted **feature importance from Random Forest model**

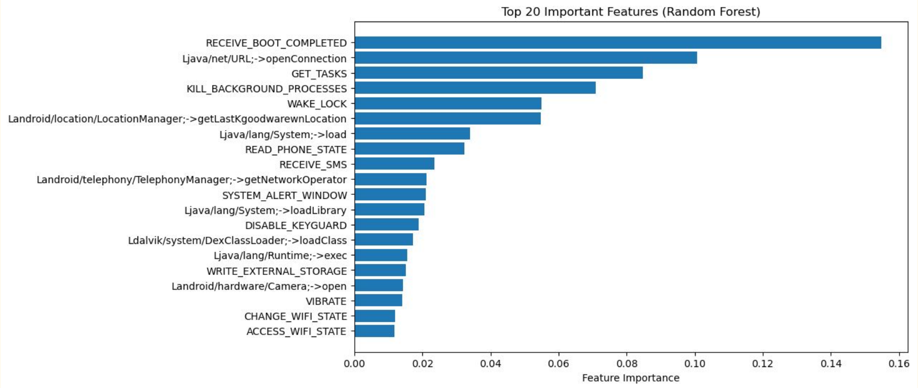
- **Top predictors driving malware classification:**

- RECEIVE_BOOT_COMPLETED → **Importance: 0.16**
  → Common in persistent malware that runs at startup
- openConnection → **Importance: 0.12**
  → Malware connecting to remote servers for payloads
- GET_TASKS → **Importance: 0.10**
  → Access to running tasks for system monitoring
- KILL_BACKGROUND_PROCESSES → **Importance: 0.08**
  → Terminate security processes to evade detection
- WAKE_LOCK → **Importance: 0.07**
  → Prevent device sleep to sustain malicious activity

These system and permission-related features **align with typical malware behavior patterns**

Real-World Impact:

- Enables **explainable AI**: Transparent model decisions
- Guides **cybersecurity policy**: Monitor high-risk permissions
- Ensures **compliance** with regulatory standards (e.g., GDPR, AI Act)



Project Insights & Deployment Readiness

- **Challenges Overcome:**
  Faced difficulty with dataset understanding and preprocessing; solved via tools like info(), head(), shape, and corr() along with ChatGPT guidance.

- **Key Findings:**

  - Addressed imbalance with **SMOTE** (applied only to training set).
  - Developed malware classifiers using **Random Forest** and **XGBoost**.
  - Achieved:
    - **Accuracy ~99.1%**
    - **F1-score > 0.97**
    - **AUC-ROC > 0.998**
  - Confusion matrices show **very few misclassifications** → strong generalization.

- **Model Interpretability:**

  - Feature importance (e.g., RECEIVE_BOOT_COMPLETED, GET_TASKS, KILL_BACKGROUND_PROCESSES) highlighted key malware indicators.
  - Insights enhance **model transparency**, aiding security experts.

- **Deployment Considerations:**
  • **Scalability** – XGBoost may need tuning for real-time use.
  • **Threshold tuning** – further reduce false negatives.
  • **Model updating** – retrain regularly to adapt to evolving malware threats.

✅ Models are **highly effective and ready for deployment**, with minor optimizations.