Kyle Kingston

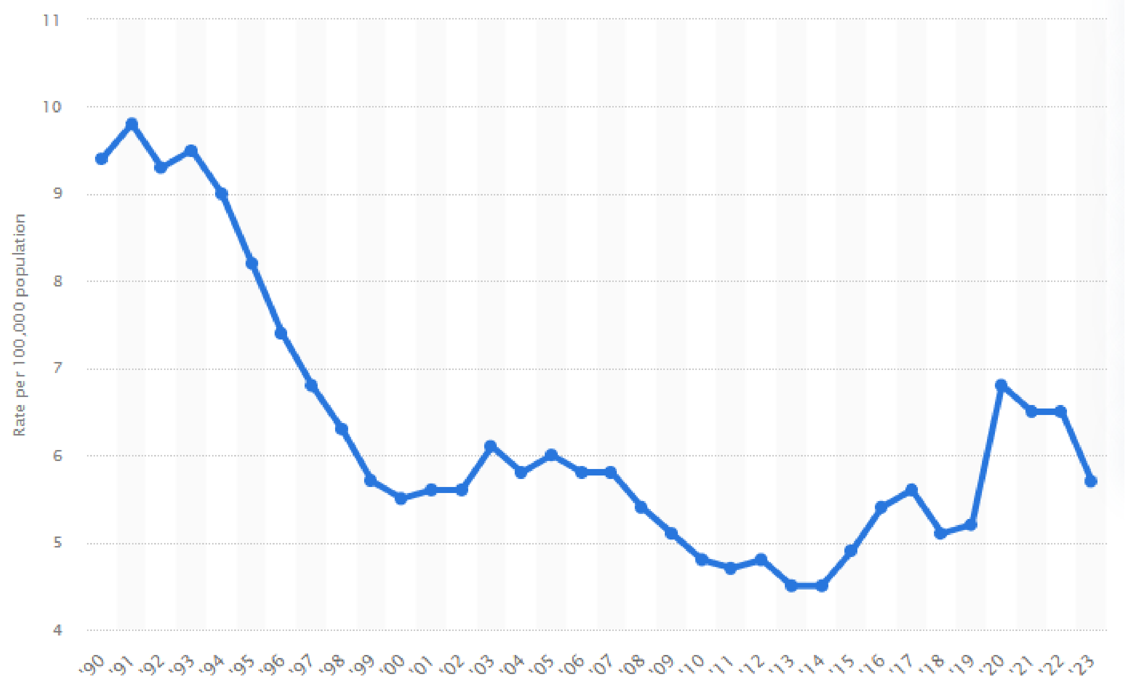
DSC630

Professor Andrew Hua

Murder stats and catch prediction

Introduction

Crime in the us has been a hot button issue at both a national, and local level. Overall murder rates appear to have been dropping until recent years where we have seen a spike.



I would like to see if we can accurately predict what crime is being committed based on various aspects of the victim. I am wanting to find if a model can accurately predict whether a murder will be solved or unsolved based on the victim and circumstance information.

Data Information

Dataset link: https://www.kaggle.com/datasets/mrayushagrawal/us-crime-dataset/data

This data set was gathered by the Us Government, Crime department in collaboration with local law enforcement agencies. This data was gathered and uploaded to Kaggle.com at the link above. The dataset consists of fields that describe the law enforcement agency who filed the report, location data for the incident, information about the incident, information about the perpetrator and information about the victim(s).

Ethical Implications

This project can potentially run into some ethical issues. While the data is public record there is still a level of privacy that is being violated by bringing these individual matters into the forefront. This evaluation will reveal some factors that make you more prone to being murdered. That can cause some people to feel uncomfortable being a potential target

Project step outline

Project steps:

1. Import data and save as data frame

2. Preform EDA
3. Visualize variables to better represent their relationship with the dataset
4. Remove variables that will cause overtraining [any perpetrator variables]
5. Divide dataset and train model
6. Evaluate model performance
7. Hyperparameter tune if necessary

For this project I plan to use a logical regression model with a target variable of the solving of the murder. This decision was based off the target variable being a binary one, whether the crime was solved or not solved. This type of target variable leans strongly toward a logistic regression model. if my logistic regression model does not meet expectations, then I will consider pivoting to a knn or neural network model.

Model Evaluation

After the model is built I will use the following evaluation metrics in order to assess the performance of the logical regression.

1. Accuracy: This will help to evaluate when our predictions are correct and when they are not based on our training and test data sets. We will be taking the mean of the cross validation scores of our model.
2. Roc Curve: This will display a graph of our true positive vs false positive rate.

3. Confusion Matrix: This is a way to compare the predictions the model makes vs the correct variable. This helps to visualize where the model is getting confused when creating its predictions
4. Classification Report: This report will provide a wide array of values that will help to evaluate the model.

Contingency plan Overview

Dataset link: https://www.kaggle.com/datasets/ikynahidwin/depression-professional-dataset

If my original project fails for some reason, I will use a separate data set to predict the outcome of depression. Essentially, this dataset has characteristics of people and records if they are diagnosed depressed or not. I would also plan to use a logical regression on this data set. The questions I would like to answer would be:

1. What factors have the largest effect on depression diagnosis
2. What demographic groups seem to have a disproportionate amount of depressed population?

The condition of utilizing this contingency are either the dataset appears to be lacking in quality or the model preforms poorly on the data even though tuning methods were applied. The last consideration for utilizing this option would be if the dataset hits some sort of memory limitation that cannot be overcome with programming methods.

The potential risks of using this contingency is the timeline of the project potentially being compromised. If the contingency is implemented during the early stages of the project this will likely not be a factor but if toward the end of the project that the contingency is used, then that may compromise the quality of the analysis if the project is rushed. I may perform eda on this dataset during the coming weeks in hopes of alleviating the concerns of time if this plan is implemented in the 11th hour.

Milestone 3

Will I be able to answer the questions I want to answer with the data I have?

Given the initial EDA that has been preformed I feel that the model will be able to discern with a relatively reliable accuracy if a crime will be solved or not. The first iteration of the model was about 70% accurate and the model should only improve when hyperparamter tuning is applied in the next iteration.

What visualizations are especially useful for explaining my data?

I find the Histograms and bar plots to be essential in understanding the data. Having an Idea of how the data is diswtributed helps to gather an understanding of the dataset as a whole. For instance, the histogram for victim count shows a large amount of 0 count offences, this raises the question of how that could be possible or what does that represent in the data. After digging further into this I have not found a conclusive answer

Do I need to adjust the data and/or driving questions?

We should be able to answer all the questions with the data on hand with the exception of the population data that we will need to pull in at a future step. Overall the questions seem to have a straight forward path for solution. Some will need to be gathered through some additional analysis.

Do I need to adjust my model/evaluation choices?

My model performs rather accurately, and I feel will yield a reasonable result for this determination. I utilized a pipeline when applying the model so it should be easy to apply additional models in order to find what yields best results. Right now, our accuracy is in the acceptable range being 72% accurate. Though I have hopes that integrating an additional model or hyperparameter tuning this model will increase the accuracy.

Are my original expectations still reasonable?

I feel the original expectations are still reasonable. The analysis seems to be coming along smoothly, and I don't foresee any insurmountable problems. If somehow issues arise, I can lean back on the backup dataset but at this point I feel like that is very unlikely to be necessary.

Milestone 4

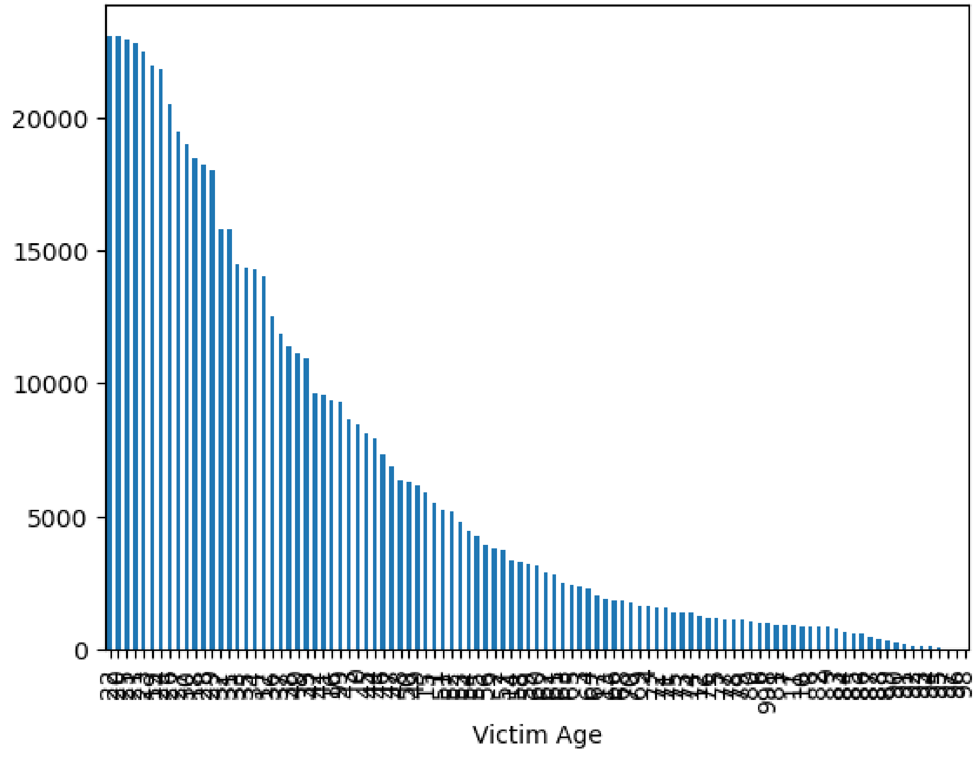
Data preparation

Data preparation was rather simple for the most part. First I evaluated any missing data to determine what the best course of action was for handing said data. This dataset was really clean in the regard and no missing values were found.

I then visualized the data using bar charts and histograms in order to form a better understanding of the data itself. This yielded a few interesting plots, the two included below are the relationship between age and incidents that clearly shows the younger you are the more likely you are to be involved in an incident. The next chart show the relation between murder wepon and occurrences.

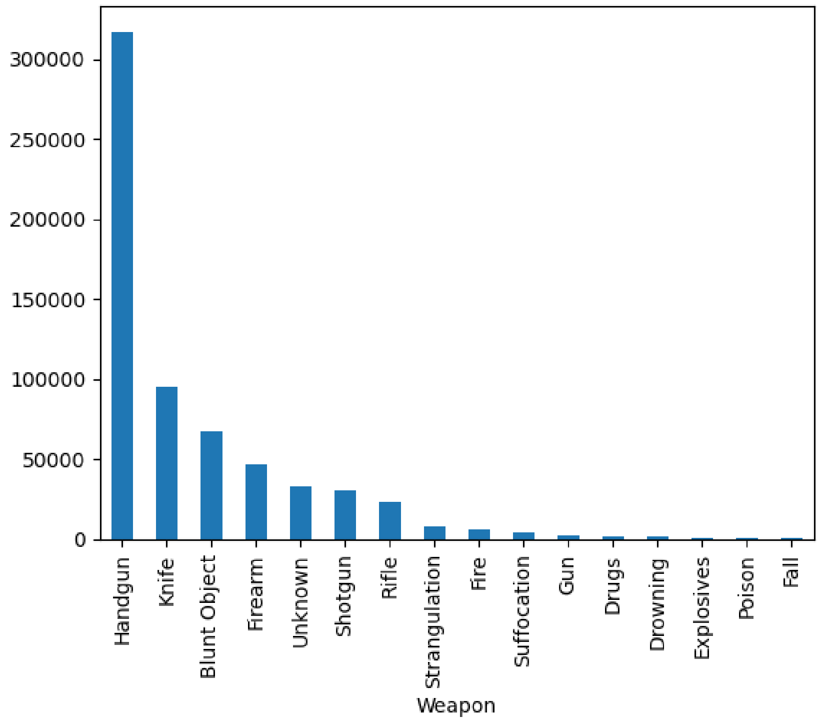
```
[15]: df['Victim Age'].value_counts().plot(kind='bar')
      #A larger amound of the victems are of a younger age
      # concider changing this to bins in the future for better readability
```

```
[15]: <Axes: xlabel='Victim Age'>
```



```
[18]: df['Weapon'].value_counts().plot(kind='bar')
      #handgun is the majority of these entries.
```

```
[18]: <Axes: xlabel='Weapon'>
```

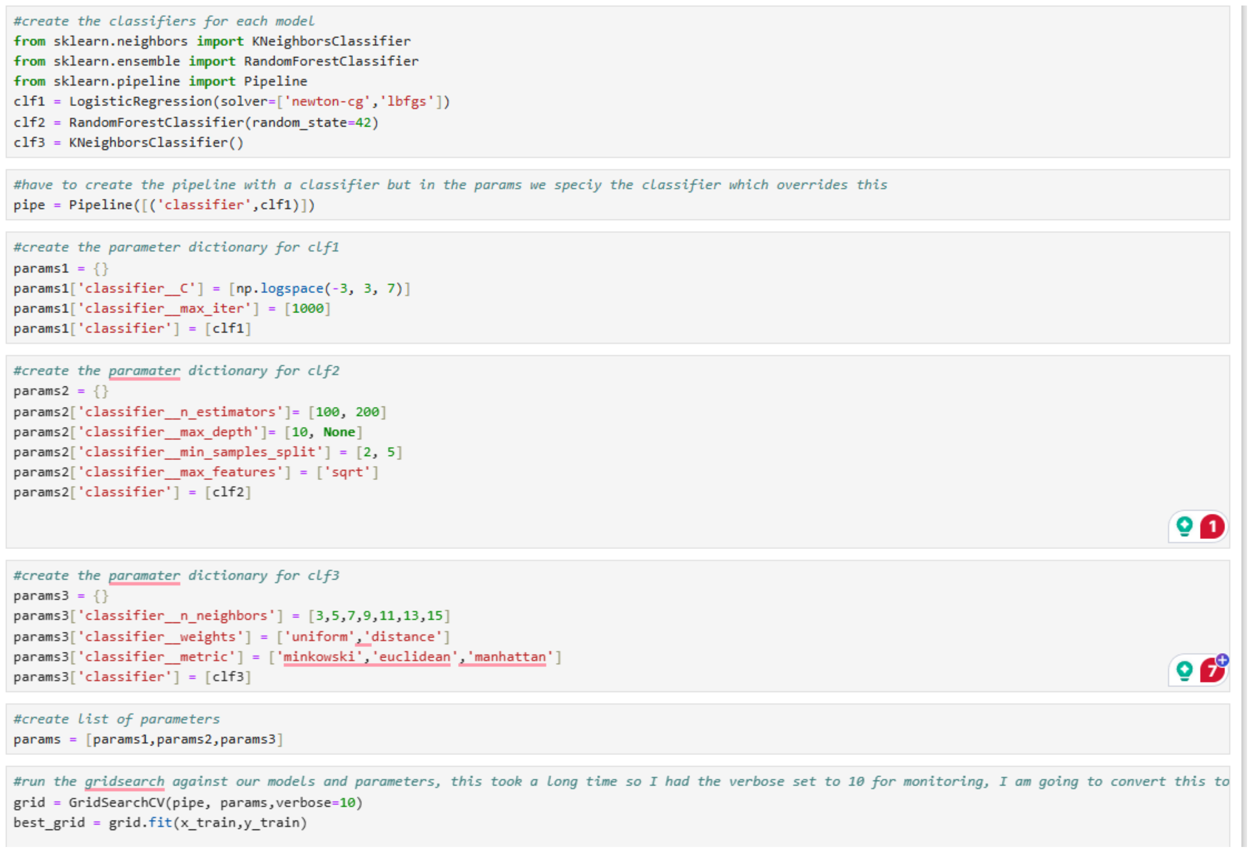


we split the dataset into our x and y values then created dummies for the categorical variables contained in the x dataset. Beyond that we scaled the data using a standard scaler, we had to use a partial fit due to memory limitations on the machine we were building the model on. Once the scaler was fit we then used it to transform out dataset.

Next we ran some pca on our dataset to see if there was a drop off of variance after a point. We were looking for a leveling out of the chart that didn't quite happen to the extent we were looking for so we did not actually apply this to our final dataset. [insert the chart here]

Now that the data is prepped, we performed a train test split and began building our logical regression model. We ran the original model without any hyperparameter tuning and got a result of ~71% accuracy. Once we had this model trained, we then created another model using hyperparameter tuning even using that method our accuracy still hovered around that 71% value. We didn't feel satisfied with that value so the pipeline was rebuilt to try knn and random forest to see if we could get a better score. [include the result metrics for this model for later comparison to the hyperparameter tuned model]

After revamping the pipeline and using a gridsearch we landed on utilizing 3 models for this evaluation: LogisticalRegression, RandomForest, and KNN. All three of these models were tuned with the parameters in the code snippit below. After running this model We saw the accuracy score increase to just over 73%.



```
#create the classifiers for each model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
clf1 = LogisticRegression(solver=['newton-cg','lbfgs'])
clf2 = RandomForestClassifier(random_state=42)
clf3 = KNeighborsClassifier()
```

```
#have to create the pipeline with a classifier but in the params we speciy the classifier which overrides this
pipe = Pipeline([('classifier',clf1)])
```

```
#create the parameter dictionary for clf1
params1 = {}
params1['classifier__C'] = [np.logspace(-3, 3, 7)]
params1['classifier__max_iter'] = [1000]
params1['classifier'] = [clf1]
```

```
#create the paramater dictionary for clf2
params2 = {}
params2['classifier__n_estimators']= [100, 200]
params2['classifier__max_depth']= [10, None]
params2['classifier__min_samples_split'] = [2, 5]
params2['classifier__max_features'] = ['sqrt']
params2['classifier'] = [clf2]
```

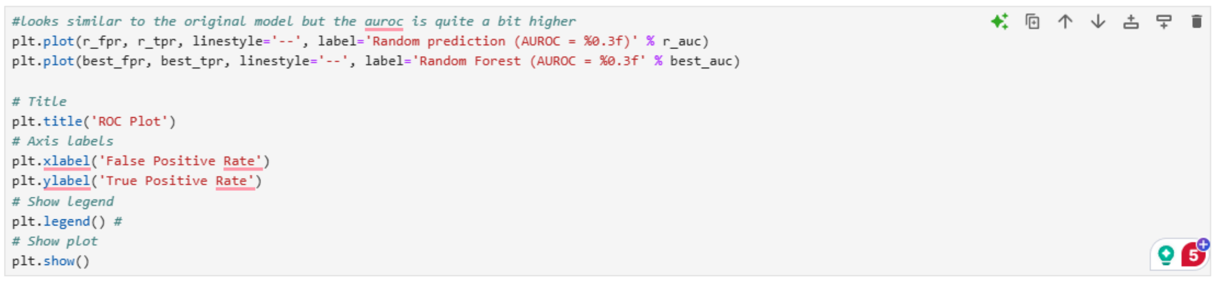
```
#create the paramater dictionary for clf3
params3 = {}
params3['classifier__n_neighbors'] = [3,5,7,9,11,13,15]
params3['classifier__weights'] = ['uniform','distance']
params3['classifier__metric'] = ['minkowski','euclidean','manhattan']
params3['classifier'] = [clf3]
```

```
#create list of parameters
params = [params1,params2,params3]
```

```
#run the gridsearch against our models and parameters, this took a long time so I had the verbose set to 10 for monitoring, I am going to convert this to
grid = GridSearchCV(pipe, params,verbose=10)
best_grid = grid.fit(x_train,y_train)
```

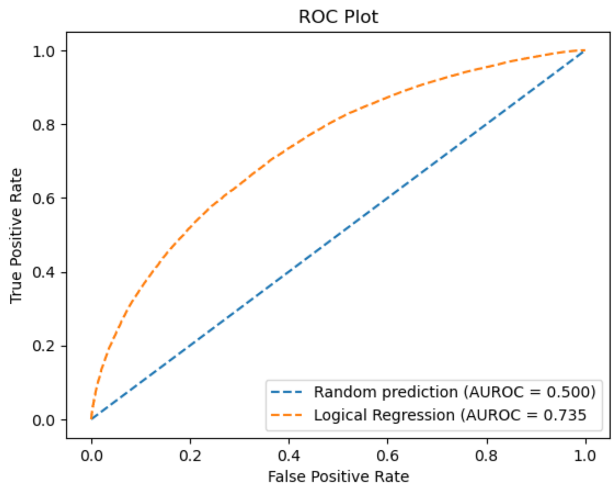
Interpretations of results

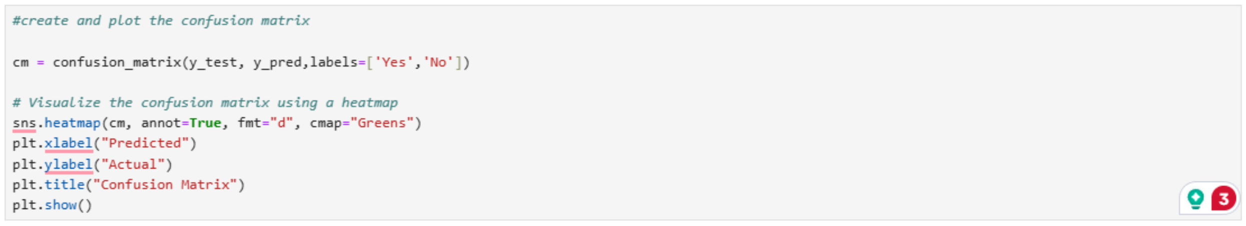
This model appears to be functioning at about 73 percent accuracy, I feel like that is a decent result. The concerning part is the number of false negatives being high. It appears the model overcompensates and leans toward murders being solvable vs not. I am considering is there is some sample curation that would produce a more balanced model and if that would in the end improve the accuracy. The hyperparameter trained model preformed a bit better then the original model showing that random forest may have been a better suite for this dataset then the logistical regression model or knn.



```
#looks similar to the original model but the auroc is quite a bit higher
plt.plot(r_fpr, r_tpr, linestyle='--', label='Random prediction (AUROC = %0.3f)' % r_auc)
plt.plot(best_fpr, best_tpr, linestyle='--', label='Random Forest (AUROC = %0.3f' % best_auc)

# Title
plt.title('ROC Plot')
# Axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
# Show plot
plt.show()
```

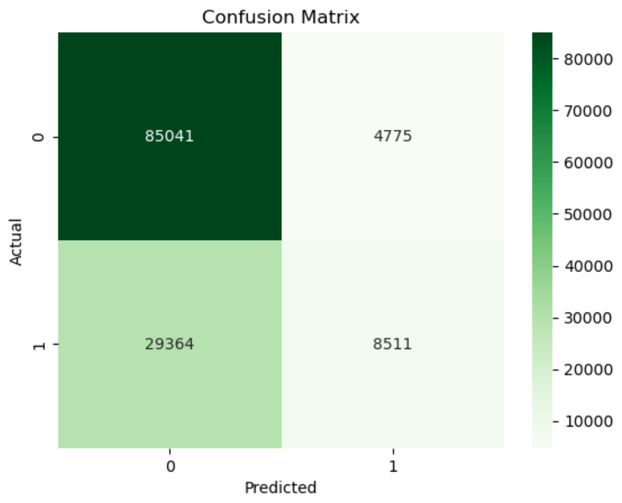


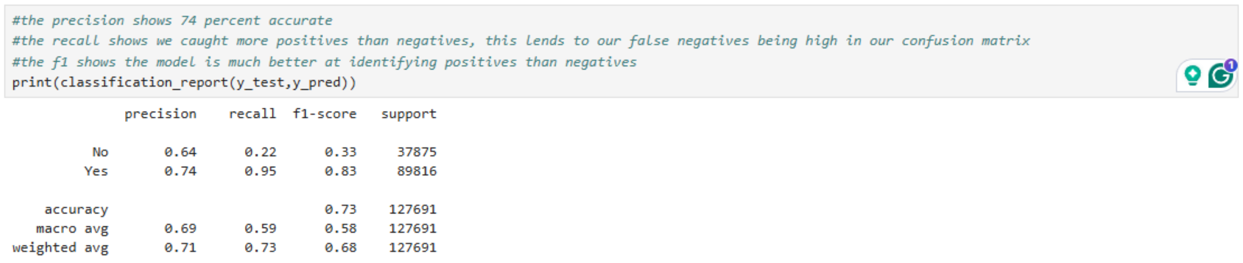


```
#create and plot the confusion matrix

cm = confusion_matrix(y_test, y_pred,labels=['Yes','No'])

# Visualize the confusion matrix using a heatmap
sns.heatmap(cm, annot=True, fmt="d", cmap="Greens")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```





```
#the precision shows 74 percent accurate
#the recall shows we caught more positives than negatives, this lends to our false negatives being high in our confusion matrix
#the f1 shows the model is much better at identifying positives than negatives
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          No       0.64      0.22      0.33     37875
         Yes       0.74      0.95      0.83     89816

    accuracy                           0.73    127691
   macro avg       0.69      0.59      0.58    127691
weighted avg       0.71      0.73      0.68    127691
```

Conclusion

Murder has and will continue to be an issue in this country and around the world. I would like to see models like this to help determine if a murder is solvable based on the circumstances of the murder and hopefully isolate why those feature seem to aid in the solving of those particular incidents. If we can isolate patters that lend to the solving of these cases hopefully we can leave less crime unsolved and result in less re-occurrence of events.

Sources:

Agrawal, A. (2023, October 2). *US crime dataset*. Kaggle. https://www.kaggle.com/datasets/mrayushagrawal/us-crime-dataset

Jatmiko, N. A. (2024, November 20). *Depression professional dataset*. Kaggle. https://www.kaggle.com/datasets/ikynahidwin/depression-professional-dataset

Korhonen, V. (2024, November 12). *USA: Reported murder and Nonnegligent Manslaughter Rate 2023*. Statista. https://www.statista.com/statistics/191223/reported-murder-and-nonnegligent-manslaughter-rate-in-the-us-since-1990/