Dataset 1:

Analysis

The confusion matrix tells us how well the model is predicting each class. For example, the model successfully predicted 437 instances of Class 1 as Class 1, but it had 6 instances of Class 1 miss-classified as Class 2. The same pattern is repeated for all the other classes, indicating where the model is doing well and where it is making an error. It also displays class-specific challenges in the matrix, misclassifying for Class 5, which is scarcely represented by examples and could be leading to poor performance by this model.

Classification Report:

The classification report includes precision, recall, and the F1 score for each of the classes, as given below:.

Class 1: Model precision was 0.89, meaning 89% of the instances posited as being in Class 1 by the model were actually in Class 1. A recall of 0.93 implies that 93% of real instances of Class 1 were well identified. The same F1-score of 0.91 gives a better tradeoff between model precision and recall.

Class 2: The high performing, high-scored class has high precision—0.98—and recall—0.98, which brings a very high F1-score of 0.98.

Class 3: The model performance was really very good, with a precision of 0.99 and recall of 1.00, thus giving an F1-score of 1.00.

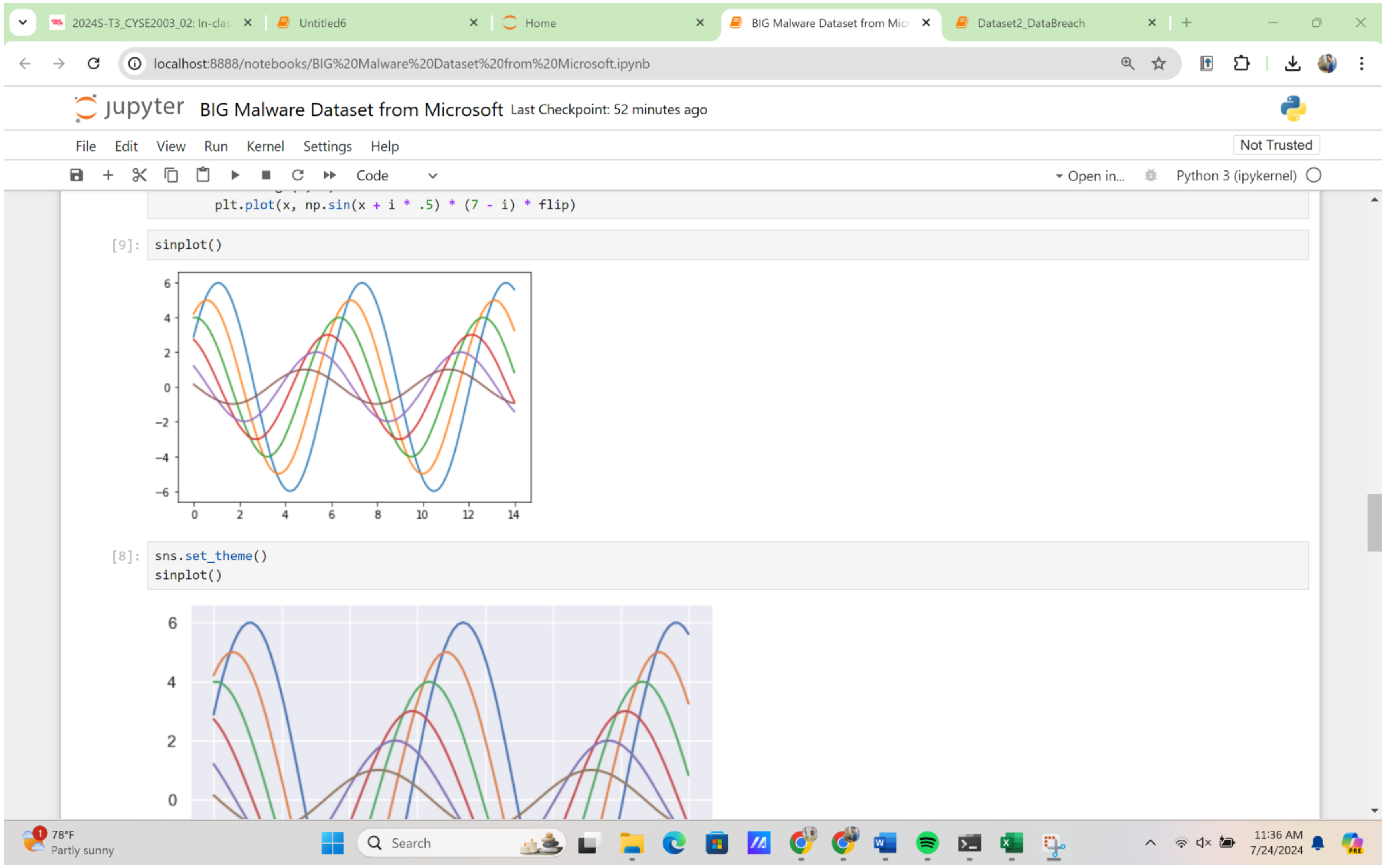
Class 4 with a precision of 0.92 and a recall of 0.99 has an F1 score of 0.95, which is very high, hence showing good performance. Class 5: The model performance on class 5 is pretty poor, with a precision of 0.40, recall of 0.18, and F1-score of 0.25; hence, the model does not perform very well in correctly predicting class 5.

Classes 6, 7, 8, and 9 described a high precision, recall, and F1-scores, which means that the model went well across these categories.

Overall Performance: On the other hand, general accuracy reaches 95%, which is indicative of strong performance across the dataset. The remaining performance measures are: macro average F1-score, 0.86, indicating that most of them are generally well performed and averaging across all classes, treating each class equally regardless of its frequency; weighted average F1-score, 0.95, showing how effective the model is when considering the distribution of the instances across different classes. The overall performance of the logistic regression model is brilliant, but it does very poorly in Class 5 due to the fact that this class is grossly underrepresented in the database.

On the subject of classification, if one is told that "Class 0 contains no positive samples, Class 1 does, and Class 2 does," then there is a serious problem either with the dataset or the way in which testing has been conducted. More specifically, Class 0 has no instances that are positive labeled in the test set, meaning there are no samples for this class that could be appraised as true recognized positives. This absence precludes the computation of some performance metrics—particularly, the ROC curve and AUC for Class 0—since these require at least some positive samples to quantify how well a model is distinguishing between positive and negative cases. On the other hand, Classes 1 and 2 have positive samples to calculate these metrics and, therefore, indicate how accurately the model is classifying instances of these classes. A scenario such as this suggests a possible imbalance, or there could be an issue with the manner in which the data distribution has been designed—that is, Class 0 is underrepresented or not part of the test data.

Screenshot



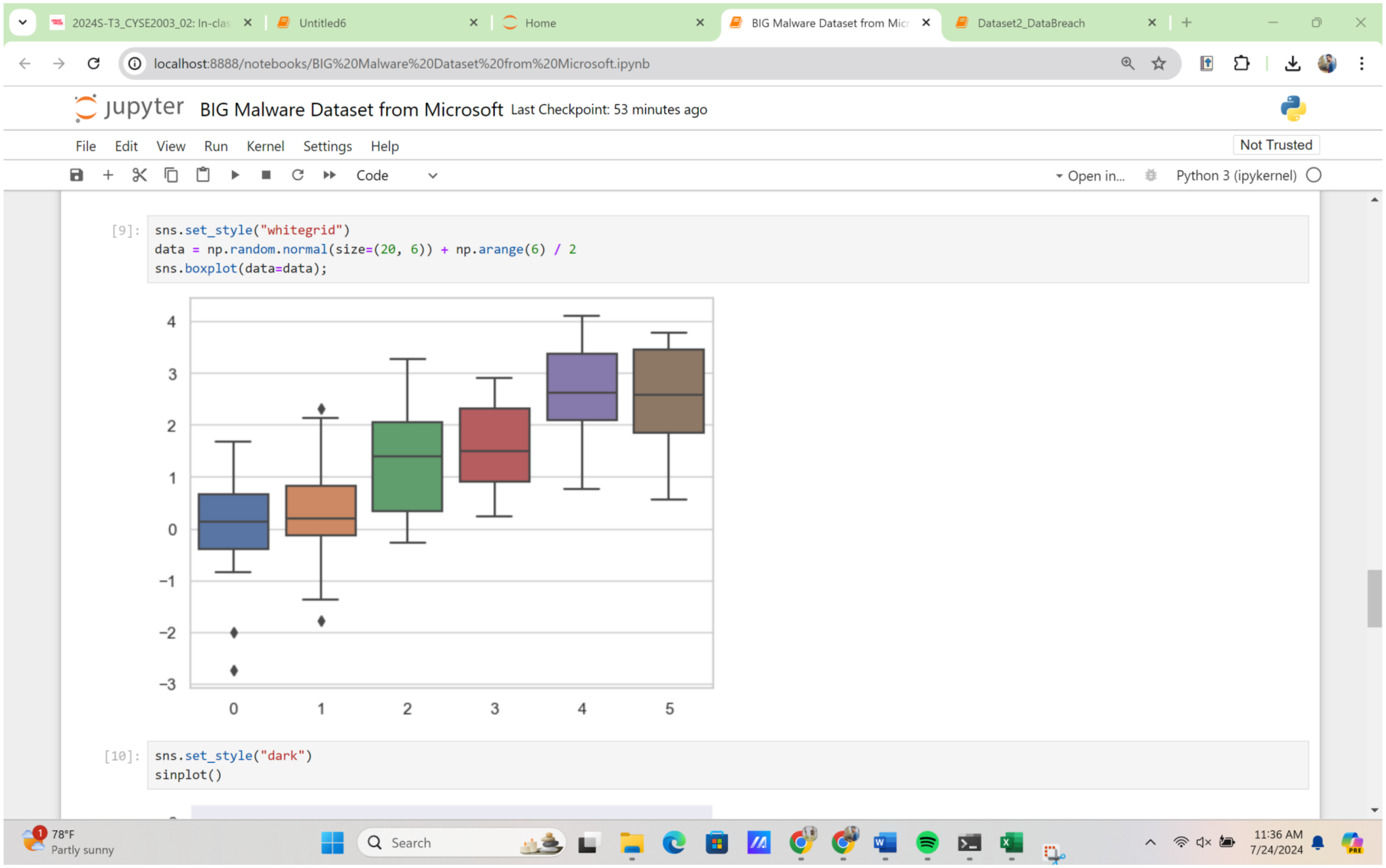
localhost:8888/notebooks/BIG%20Malware%20Dataset%20from%20Microsoft.ipynb

BIG Malware Dataset from Microsoft Last Checkpoint: 52 minutes ago

```
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
```

```
[9]: sinplot()
```

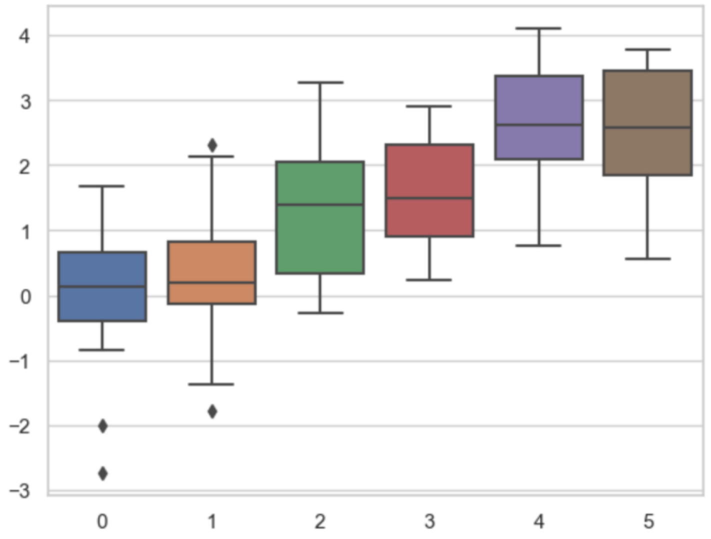
```
[8]: sns.set_theme()
     sinplot()
```



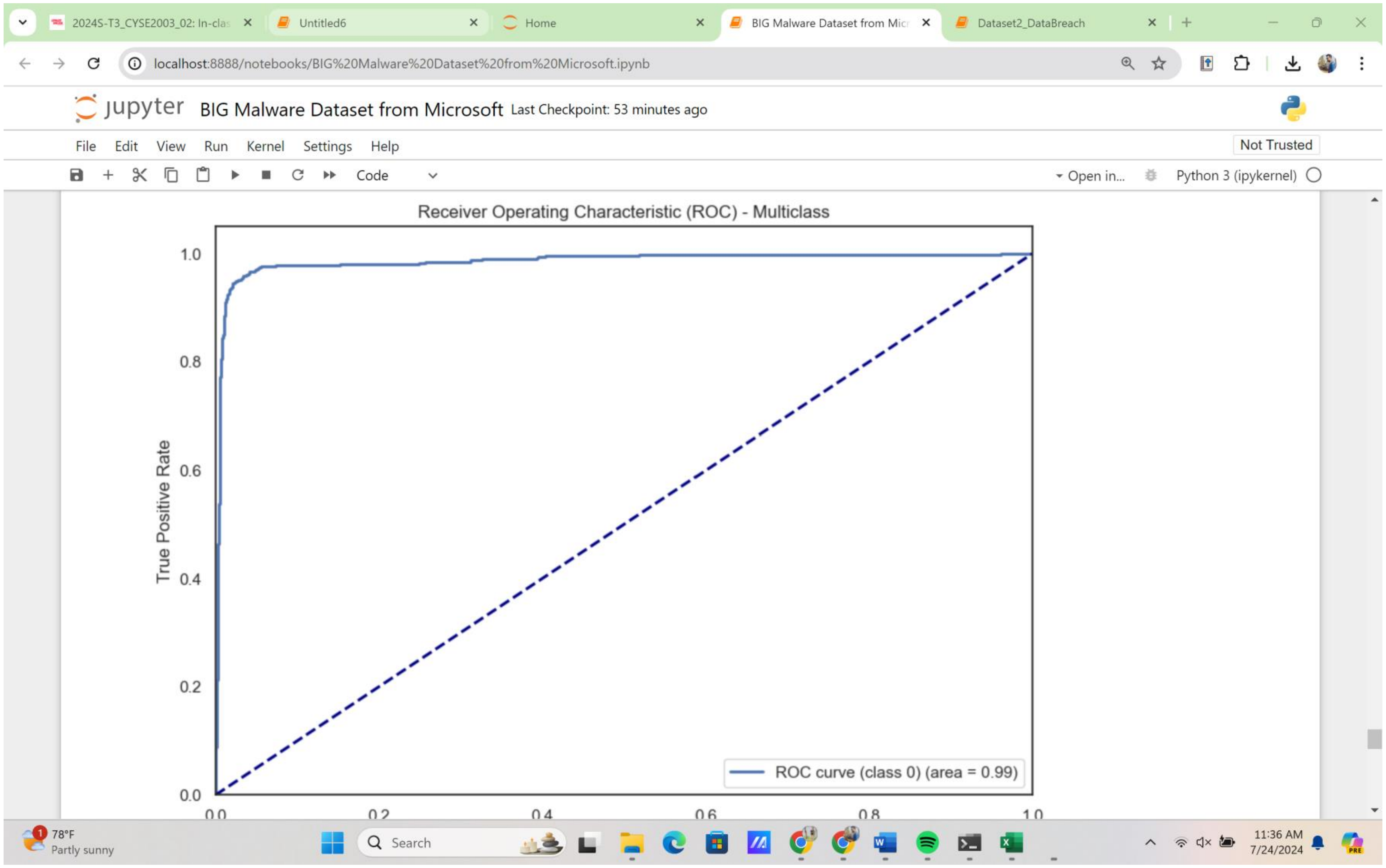
localhost:8888/notebooks/BIG%20Malware%20Dataset%20from%20Microsoft.ipynb

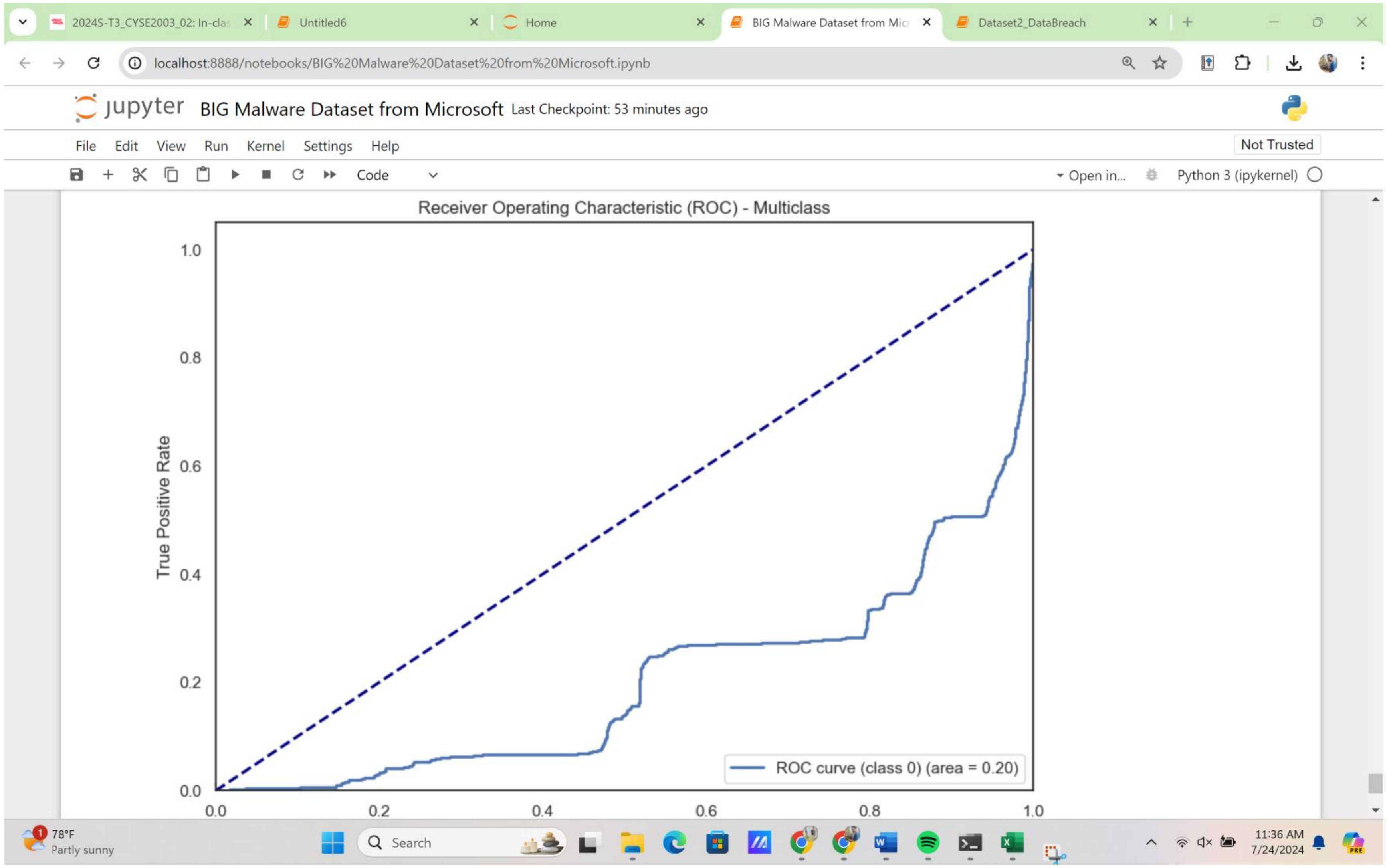
BIG Malware Dataset from Microsoft Last Checkpoint: 53 minutes ago

```
[9]: sns.set_style("whitegrid")
     data = np.random.normal(size=(20, 6)) + np.arange(6) / 2
     sns.boxplot(data=data);
```



```
[10]: sns.set_style("dark")
      sinplot()
```


2024S-T3_CYSE2003_02: In-clas
Untitled6
Home
BIG Malware Dataset from Mic
Dataset2_DataBreach
localhost:8888/notebooks/BIG%20Malware%20Dataset%20from%20Microsoft.ipynb
jupyter
BIG Malware Dataset from Microsoft
Last Checkpoint: 53 minutes ago
File
Edit
View
Run
Kernel
Settings
Help
Not Trusted
Code
Open in...
Python 3 (ipykernel)
Receiver Operating Characteristic (ROC) - Multiclass
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (class 0) (area = 0.99)
78°F
Partly sunny
Search
11:36 AM
7/24/2024


2024S-T3_CYSE2003_02: In-clas
Untitled6
Home
BIG Malware Dataset from Mic
Dataset2_DataBreach
localhost:8888/notebooks/BIG%20Malware%20Dataset%20from%20Microsoft.ipynb
jupyter
BIG Malware Dataset from Microsoft
Last Checkpoint: 53 minutes ago
File
Edit
View
Run
Kernel
Settings
Help
Not Trusted
Code
Open in...
Python 3 (ipykernel)
Receiver Operating Characteristic (ROC) - Multiclass
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (class 0) (area = 0.20)
78°F
Partly sunny
Search
11:36 AM
7/24/2024

Dataset 2: Analysis

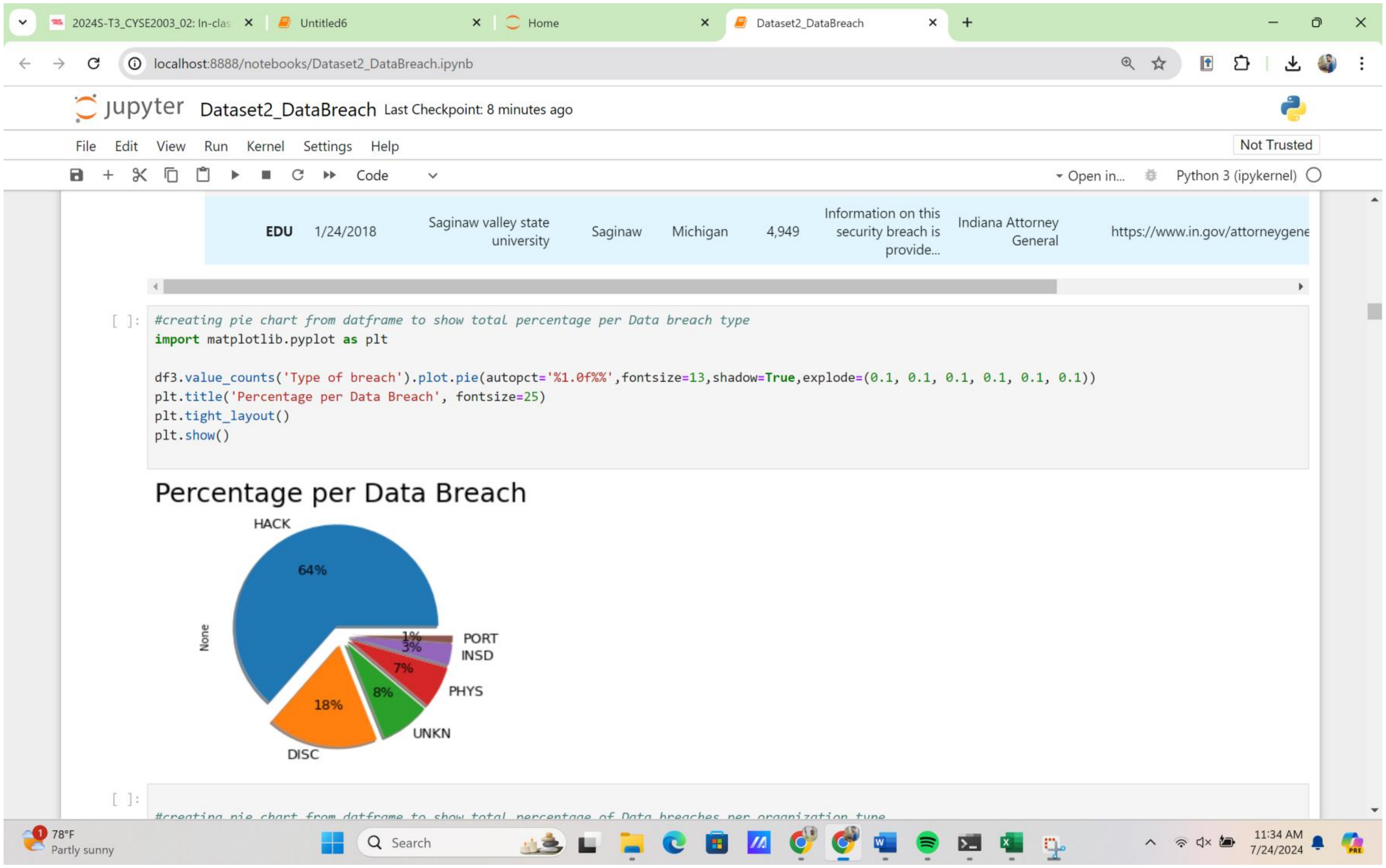
The pie chart showing the percentage of data breaches by the organization's type has indicated a high concentration of breaches in the healthcare sector, that is MED and business organizations-BSO. This is an important sign or identification of vulnerability of health and various business sectors due to the sensitivity of medical data and the diversity of businesses covered under BSO. These high frequencies in breach of these sectors suggest increased security measures and prevention strategies that are more specific to the risk of healthcare data and diversified business operations.

The following bar chart provides the total number of incidents by breach type. From the chart, it is clear that the hacking category, represented as HACK, is the clear leader. It means that organizations are often attacked from the outside and infected with malware more than any other kind of breach.

Indeed, the occurrence of hacking incidents only serves to illustrate the need for good cybersecurity protocols as well as defensive measures that prevent attacks coming from outside. Net security hardening, periodic security updates, and vulnerability assessment can reduce the risk of such an attack.

Pie Chart for the Top 5 States with Most Hacks In that direction, the pie chart showing associated breach incidents in regard to the top 5 states with most hacks has presented a geographical context. Targeting regional cyber security efforts, this spatial distribution has the potential ability of effectively prioritizing resources towards better protection of data in states associated with higher breaches. Factors specific to such states which contribute more towards higher breach rates would be identified and helped a lot in the improvement of overall data protection. Finally, the word cloud arising from cleaned text data helps to pick on the most frequently occurring words related to data breaches. From this visualization, common terminology as well as themes associated with breaches that help in understanding what is being said of the language and nature of incidents reported can be realized. Patterns or common recurring issues can also be realized which could inform future research or preventive measures.

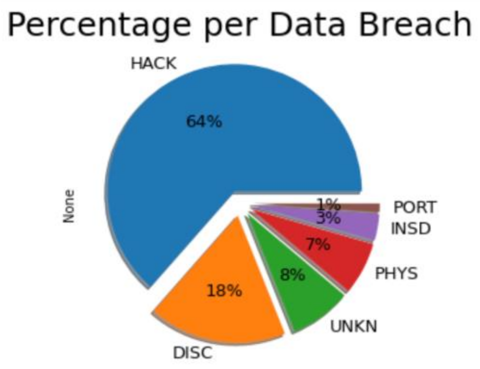
Screenshots:



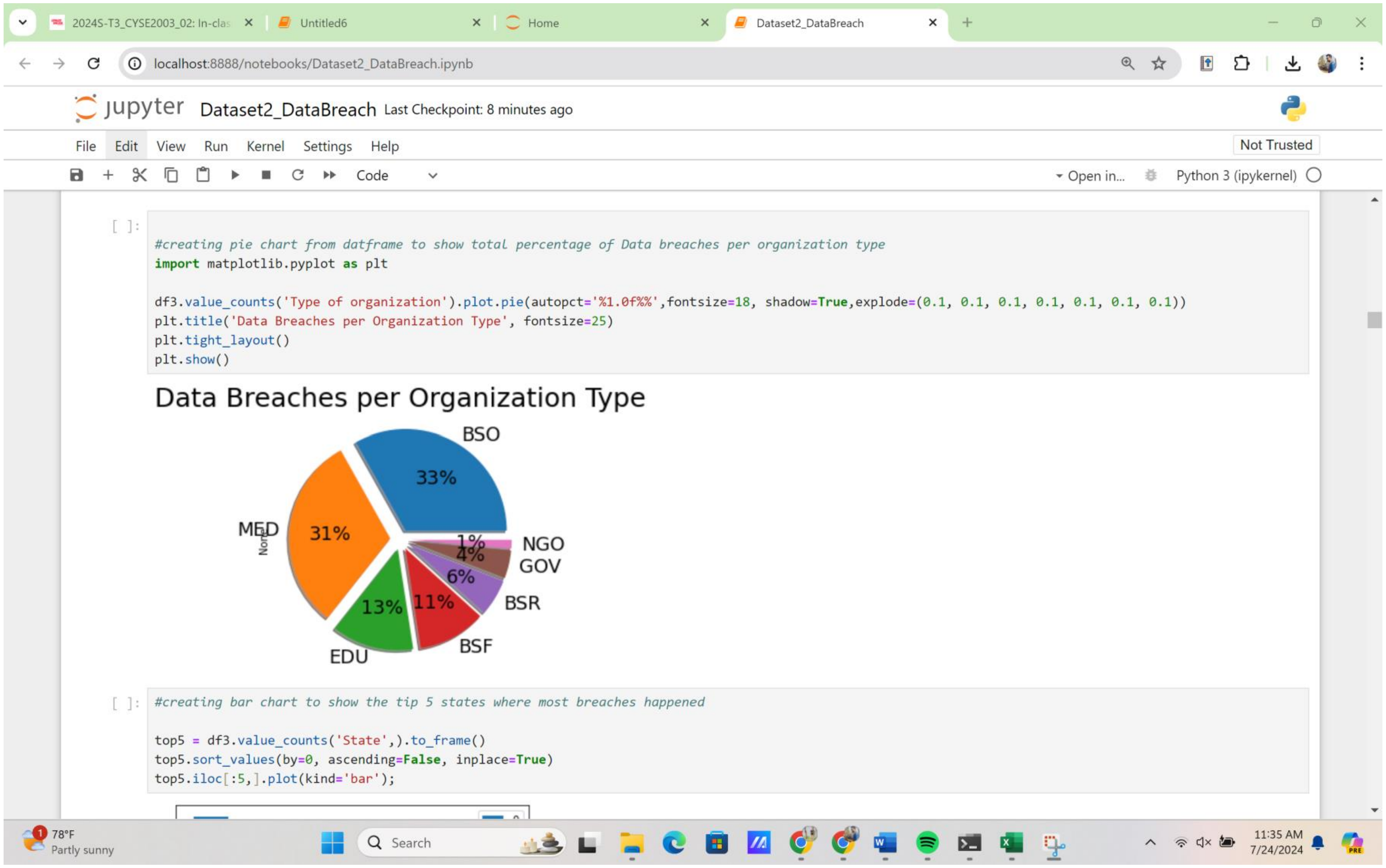
EDU	1/24/2018	Saginaw valley state university	Saginaw	Michigan	4,949	Information on this security breach is provide...	Indiana Attorney General	https://www.in.gov/attorneygene

```
[ ]: #creating pie chart from datframe to show total percentage per Data breach type
     import matplotlib.pyplot as plt

     df3.value_counts('Type of breach').plot.pie(autopct='%1.0f%%',fontsize=13,shadow=True,explode=(0.1, 0.1, 0.1, 0.1, 0.1, 0.1))
     plt.title('Percentage per Data Breach', fontsize=25)
     plt.tight_layout()
     plt.show()
```



```
[ ]: #creating pie chart from datframe to show total percentage of Data breaches per organization type
```

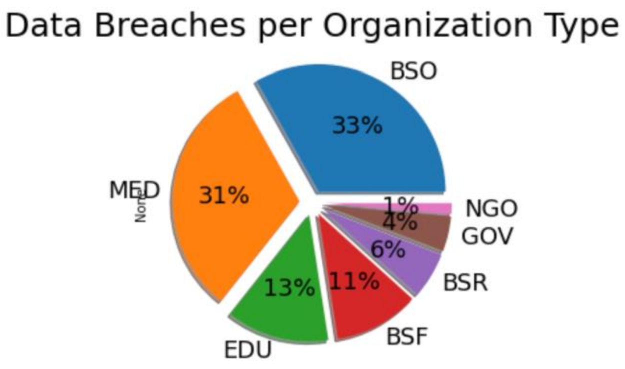


localhost:8888/notebooks/Dataset2_DataBreach.ipynb

Dataset2_DataBreach Last Checkpoint: 8 minutes ago

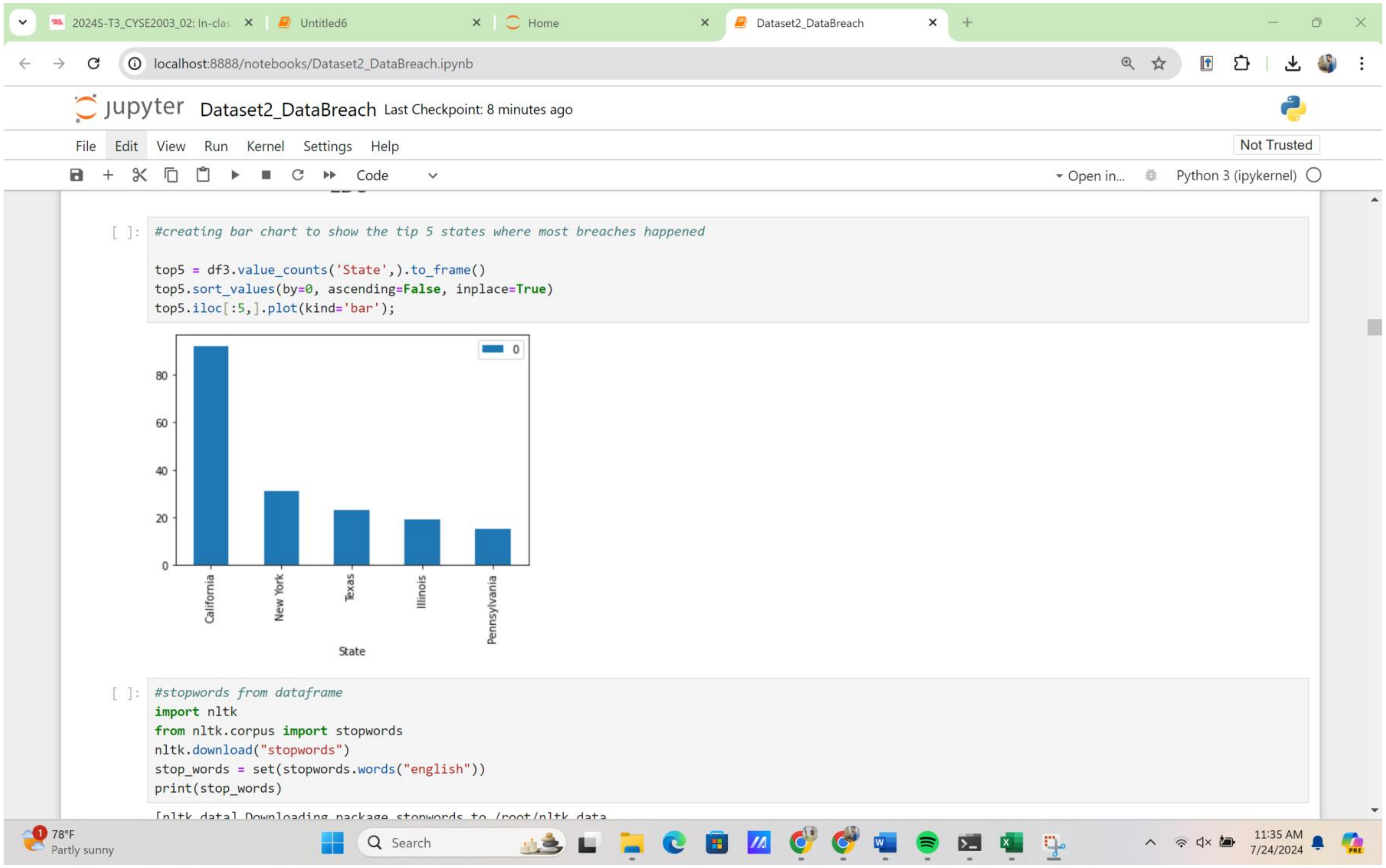
```
#creating pie chart from datframe to show total percentage of Data breaches per organization type
import matplotlib.pyplot as plt

df3.value_counts('Type of organization').plot.pie(autopct='%1.0f%%',fontsize=18, shadow=True,explode=(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1))
plt.title('Data Breaches per Organization Type', fontsize=25)
plt.tight_layout()
plt.show()
```



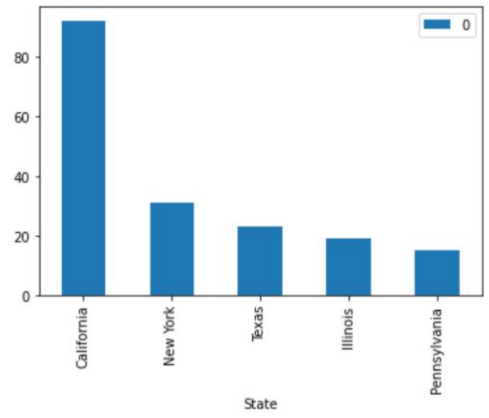
```
#creating bar chart to show the tip 5 states where most breaches happened

top5 = df3.value_counts('State',).to_frame()
top5.sort_values(by=0, ascending=False, inplace=True)
top5.iloc[:5,].plot(kind='bar');
```


2024S-T3_CYSE2003_02: In-clas
Untitled6
Home
Dataset2_DataBreach
localhost:8888/notebooks/Dataset2_DataBreach.ipynb
jupyter Dataset2_DataBreach Last Checkpoint: 8 minutes ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
Open in...
Python 3 (ipykernel)
78°F
Partly sunny
Search
11:35 AM
7/24/2024

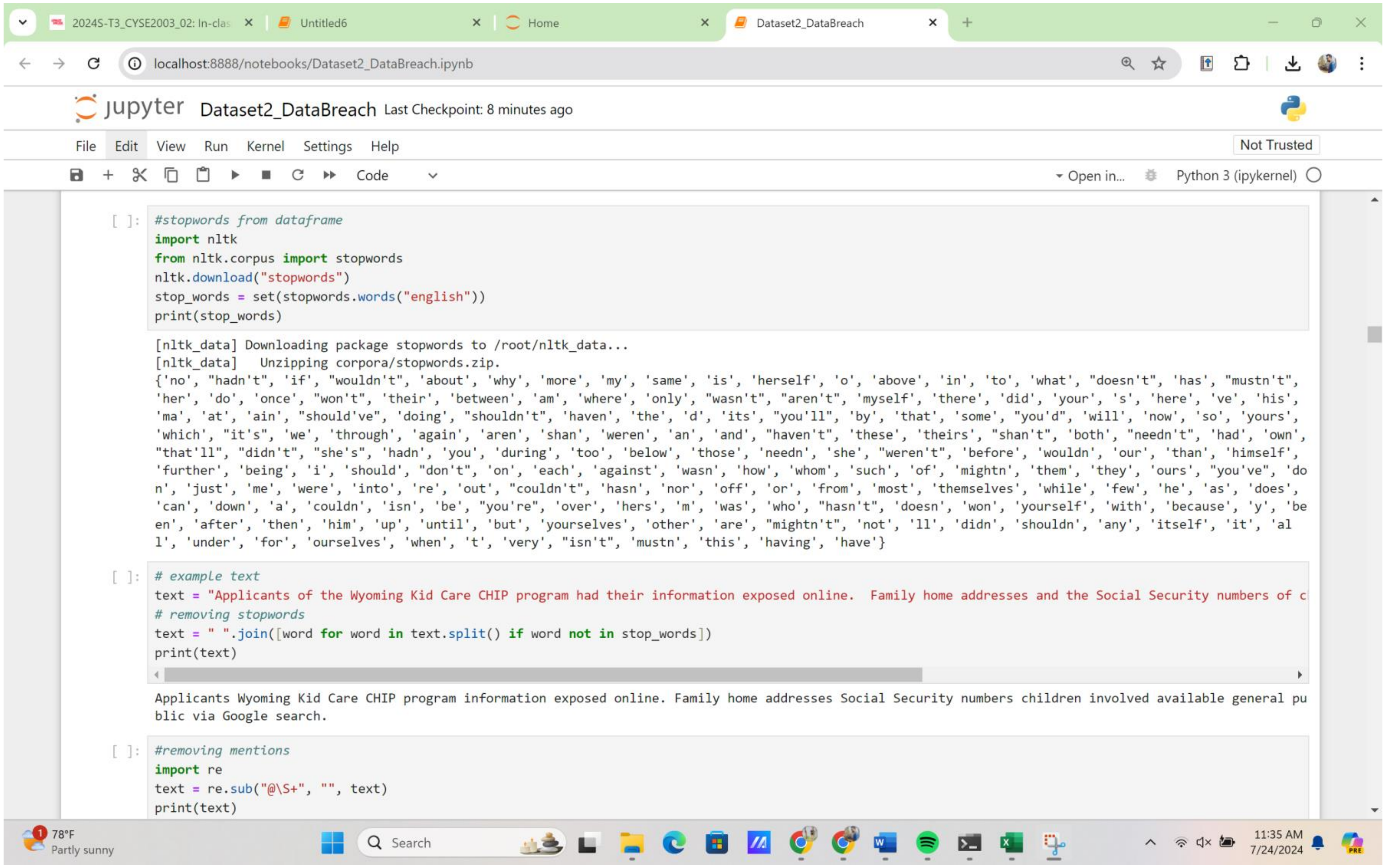
```
[ ]: #creating bar chart to show the tip 5 states where most breaches happened

     top5 = df3.value_counts('State',).to_frame()
     top5.sort_values(by=0, ascending=False, inplace=True)
     top5.iloc[:5,].plot(kind='bar');
```



```
[ ]: #stopwords from dataframe
     import nltk
     from nltk.corpus import stopwords
     nltk.download("stopwords")
     stop_words = set(stopwords.words("english"))
     print(stop_words)

     [nltk_data] Downloading package stopwords to /root/nltk_data
```


2024S-T3_CYSE2003_02: In-clas
Untitled6
Home
Dataset2_DataBreach
localhost:8888/notebooks/Dataset2_DataBreach.ipynb
jupyter Dataset2_DataBreach Last Checkpoint: 8 minutes ago
File Edit View Run Kernel Settings Help
Not Trusted
Code
Open in...
Python 3 (ipykernel)
78°F
Partly sunny
Search
11:35 AM
7/24/2024

```
[ ]: #stopwords from dataframe
     import nltk
     from nltk.corpus import stopwords
     nltk.download("stopwords")
     stop_words = set(stopwords.words("english"))
     print(stop_words)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
{'no', "hadn't", 'if', "wouldn't", 'about', 'why', 'more', 'my', 'same', 'is', 'herself', 'o', 'above', 'in', 'to', 'what', "doesn't", 'has', "mustn't",
'her', 'do', 'once', "won't", 'their', 'between', 'am', 'where', 'only', "wasn't", "aren't", 'myself', 'there', 'did', 'your', 's', 'here', 've', 'his',
'ma', 'at', 'ain', "should've", 'doing', "shouldn't", 'haven', 'the', 'd', 'its', "you'll", 'by', 'that', 'some', "you'd", 'will', 'now', 'so', 'yours',
'which', "it's", 'we', 'through', 'again', 'aren', 'shan', 'weren', 'an', 'and', "haven't", 'these', 'theirs', "shan't", 'both', "needn't", 'had', 'own',
"that'll", "didn't", "she's", 'hadn', 'you', 'during', 'too', 'below', 'those', 'needn', 'she', "weren't", 'before', 'wouldn', 'our', 'than', 'himself',
'further', 'being', 'i', 'should', "don't", 'on', 'each', 'against', 'wasn', 'how', 'whom', 'such', 'of', 'mightn', 'them', 'they', 'ours', "you've", 'do
n', 'just', 'me', 'were', 'into', 're', 'out', "couldn't", 'hasn', 'nor', 'off', 'or', 'from', 'most', 'themselves', 'while', 'few', 'he', 'as', 'does',
'can', 'down', 'a', 'couldn', 'isn', 'be', "you're", 'over', 'hers', 'm', 'was', 'who', "hasn't", 'doesn', 'won', 'yourself', 'with', 'because', 'y', 'be
en', 'after', 'then', 'him', 'up', 'until', 'but', 'yourselves', 'other', 'are', "mightn't", 'not', 'll', 'didn', 'shouldn', 'any', 'itself', 'it', 'al
l', 'under', 'for', 'ourselves', 'when', 't', 'very', "isn't", 'mustn', 'this', 'having', 'have'}
```

```
[ ]: # example text
     text = "Applicants of the Wyoming Kid Care CHIP program had their information exposed online.  Family home addresses and the Social Security numbers of c
     # removing stopwords
     text = " ".join([word for word in text.split() if word not in stop_words])
     print(text)
```

```
Applicants Wyoming Kid Care CHIP program information exposed online. Family home addresses Social Security numbers children involved available general pu
blic via Google search.
```

```
[ ]: #removing mentions
     import re
     text = re.sub("@\S+", "", text)
     print(text)
```