welcome-to-colaboratory

April 2, 2024

Welcome to Colab!

(New) Try the Gemini API

Generate a Gemini API key

Create a marketing campaign from a product sketch of a Jet Backpack

Gemini API: Quickstart with Python

Gemini API code sample

Compare Gemini with ChatGPT

More notebooks

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.

[ ]:

What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with - Zero configuration required - Access to GPUs free of charge - Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

0.1 Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

[ ]:
```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

[ ]:
```
86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

[ ]:
```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

[ ]:
```
604800
```

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see jupyter.org.

0.2 Data science

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

[ ]:
```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To learn more about importing data, and how Colab can be used for data science, see the links below under Working with Data.

0.3 Machine learning

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

Colab is used extensively in the machine learning community with applications including: - Getting started with TensorFlow - Developing and training neural networks - Experimenting with TPUs - Disseminating AI research - Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

0.4 More Resources

0.4.1 Working with Notebooks in Colab

- Overview of Colaboratory
- Guide to Markdown
- Importing libraries and installing dependencies
- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets

Working with Data

- Loading data: Drive, Sheets, and Google Cloud Storage
- Charts: visualizing data
- Getting started with BigQuery

0.4.2 Machine Learning Crash Course

These are a few of the notebooks from Google's online Machine Learning course. See the full course website for more. - Intro to Pandas DataFrame - Linear regression with tf.keras using synthetic data

Using Accelerated Hardware

- TensorFlow with GPUs
- TensorFlow with TPUs

0.4.3 Featured examples

- NeMo Voice Swap: Use Nvidia's NeMo conversational AI Toolkit to swap a voice in an audio fragment with a computer generated one.
- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB movie reviews as either *positive* or *negative*.
- Style Transfer: Use deep learning to transfer style between images.

- Multilingual Universal Sentence Encoder Q&A: Use a machine learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.

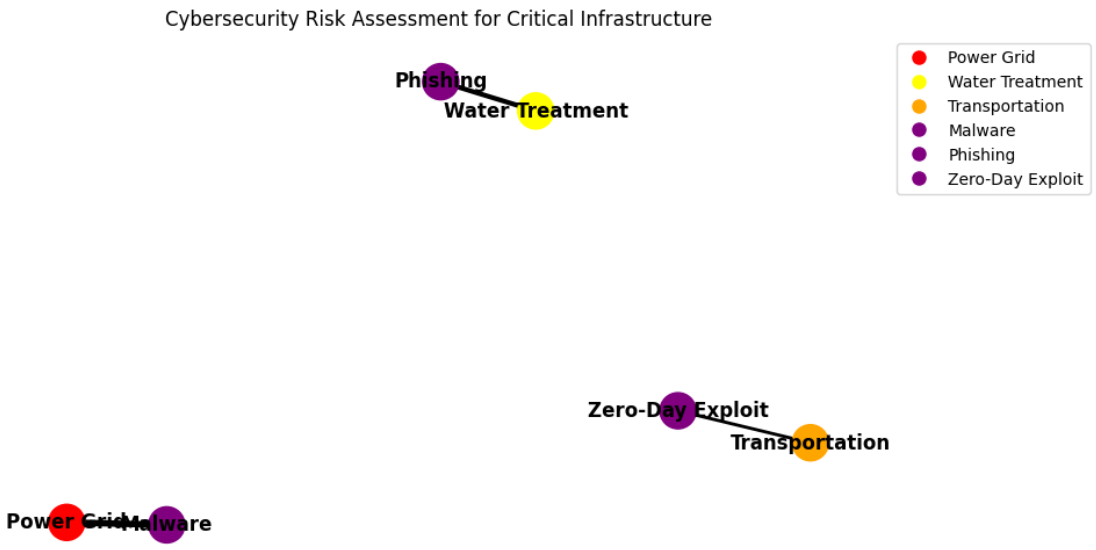
[1]:
```
import numpy as np

class BayesianNetwork:
    def __init__(self, nodes, edges):
        self.nodes = nodes
        self.edges = edges
        self.probabilities = {}

    def set_probability(self, node, parents, probability):
        key = (node, tuple(parents))
        self.probabilities[key] = probability

    def get_probability(self, node, parents, value):
        key = (node, tuple(parents))
        return self.probabilities.get(key, {}).get(value, 0.5) # Default to 0.5␣
 ↪if probability is not specified

    def calculate_probability(self, node, evidence):
        parents = [edge[0] for edge in self.edges if edge[1] == node]
        probabilities = [self.get_probability(node, parents, True) if␣
 ↪evidence[parent] else self.get_probability(node, parents, False) for parent␣
 ↪in parents]
        return np.prod(probabilities) if probabilities else 0.5

def main():
    # Define nodes and edges of the Bayesian network
    nodes = ['Attack', 'Vulnerability', 'ThreatIntel', 'Risk']
    edges = [('Attack', 'Risk'), ('Vulnerability', 'Risk'), ('ThreatIntel',␣
 ↪'Risk')]

    # Initialize the Bayesian network
    bayesian_network = BayesianNetwork(nodes, edges)

    # Set probabilities for each node and its parents
    bayesian_network.set_probability('Risk', ['Attack', 'Vulnerability',␣
 ↪'ThreatIntel'], {True: 0.8, False: 0.2})
    bayesian_network.set_probability('Attack', [], {True: 0.1, False: 0.9})
    bayesian_network.set_probability('Vulnerability', [], {True: 0.3, False: 0.
 ↪7})
    bayesian_network.set_probability('ThreatIntel', [], {True: 0.4, False: 0.6})

    # Predict the risk probability given evidence
    evidence = {'Attack': True, 'Vulnerability': True, 'ThreatIntel': True}
```

```
    risk_probability = bayesian_network.calculate_probability('Risk', evidence)

    print("Predicted Risk Probability:", risk_probability)

if __name__ == "__main__":
    main()
```

```
Predicted Risk Probability: 0.5120000000000001
```

[2]:
```
import networkx as nx
import matplotlib.pyplot as plt

# Define infrastructure types and threats
infrastructure = ["Power Grid", "Water Treatment", "Transportation"]
threats = ["Malware", "Phishing", "Zero-Day Exploit"]

# Create a graph object
G = nx.Graph()

# Add infrastructure nodes with vulnerability scores
for infra in infrastructure:
    G.add_node(infra, vulnerability=0.7)  # Replace 0.7 with actual score

# Add threat nodes
G.add_nodes_from(threats)

# Add edges with likelihoods (weights)
edges = [
    ("Malware", "Power Grid", 0.8),  # 80% chance of malware targeting power␣
 ↪grid
    ("Phishing", "Water Treatment", 0.6),  # 60% chance of phishing for water␣
 ↪treatment
    ("Zero-Day Exploit", "Transportation", 0.4),  # 40% chance of zero-day␣
 ↪exploit for transportation
]

G.add_weighted_edges_from(edges)

# Define node colors based on impact severity (replace with actual values)
node_colors = {"Power Grid": "red", "Water Treatment": "yellow",␣
 ↪"Transportation": "orange"}
node_colors.update({t: "purple" for t in threats})  # Purple for threats

# Create the plot
plt.figure(figsize=(10, 6))
pos = nx.spring_layout(G)
# Draw nodes
```

```
nx.draw_networkx_nodes(G, pos, node_color=[node_colors[node] for node in G.
 ↪nodes()], node_size=500)

# Draw edges
nx.draw_networkx_edges(G, pos, width=[G.edges[edge]['weight'] * 5 for edge in G.
 ↪edges()])

# Draw labels
nx.draw_networkx_labels(G, pos, font_size=12, font_weight="bold")

# Add legend for node colors
legend_labels = [plt.Line2D([0], [0], marker='o', color='w',␣
 ↪markerfacecolor=color, markersize=10, label=label)
                 for label, color in node_colors.items()]
plt.legend(handles=legend_labels, loc="upper left", bbox_to_anchor=(1, 1))

# Remove axis
plt.axis("off")

# Title
plt.title("Cybersecurity Risk Assessment for Critical Infrastructure")

# Show plot
plt.show()
```



[3]:
```
import numpy as np
import pandas as pd
```

```
import itertools
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
```

[4]:
```
np.random.seed(0)
num_samples = 1000
features = {
    'time_of_day': np.random.randint(0, 24, num_samples),
    'failed_attempts_24h': np.random.randint(0, 50, num_samples),
    'failed_attempts_1h': np.random.randint(0, 10, num_samples),
    'is_common_password': np.random.randint(0, 2, num_samples)
}
labels = np.random.randint(0, 2, num_samples)  # Binary labels (0: not attack,␣
 ↪1: attack)
data = pd.DataFrame({**features, 'is_attack': labels})
```

[5]:
```
X = data.drop(columns=['is_attack'])
y = data['is_attack']
```

[6]:
```
predictions = rf_classifier.predict(X_test)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-b5111ce2e918> in <cell line: 1>()
----> 1 predictions = rf_classifier.predict(X_test)

NameError: name 'rf_classifier' is not defined
```

[7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[8]:
```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
```

[8]:
```
RandomForestClassifier(random_state=42)
```

[9]:
```
predictions = rf_classifier.predict(X_test)
```

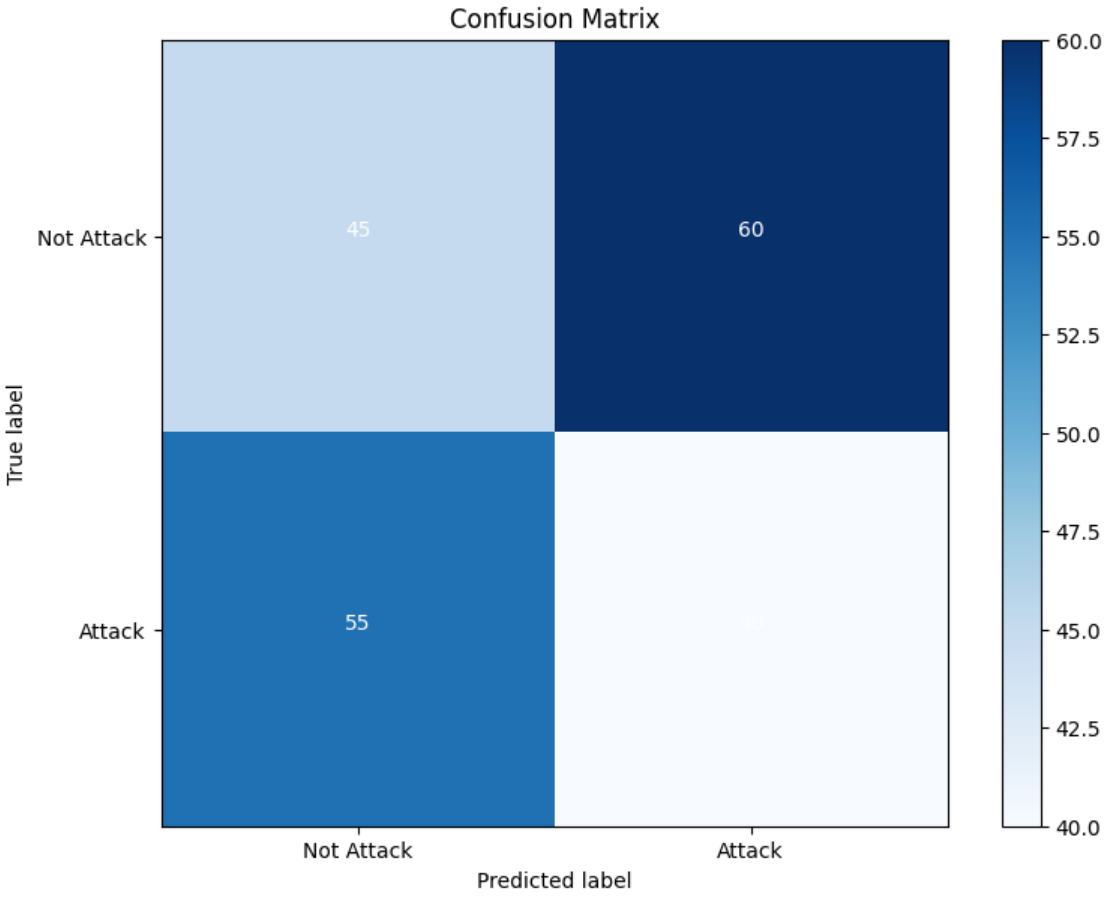
[10]:
```
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.425
```

[11]:
```
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[45 60]
 [55 40]]
```

[12]:
```
plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, cmap='Blues', interpolation='nearest')
plt.title('Confusion Matrix')
plt.colorbar()

classes = ['Not Attack', 'Attack']
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes)
plt.yticks(tick_marks, classes)

thresh = conf_matrix.max() / 2.
for i, j in itertools.product(range(conf_matrix.shape[0]), range(conf_matrix.
 ↪shape[1])):
    plt.text(j, i, format(conf_matrix[i, j], 'd'),
             horizontalalignment="center",
             color="white" if conf_matrix[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```


Confusion Matrix
Not Attack
Attack
True label
45
60
55
Not Attack
Attack
Predicted label
60.0
57.5
55.0
52.5
50.0
47.5
45.0
42.5
40.0