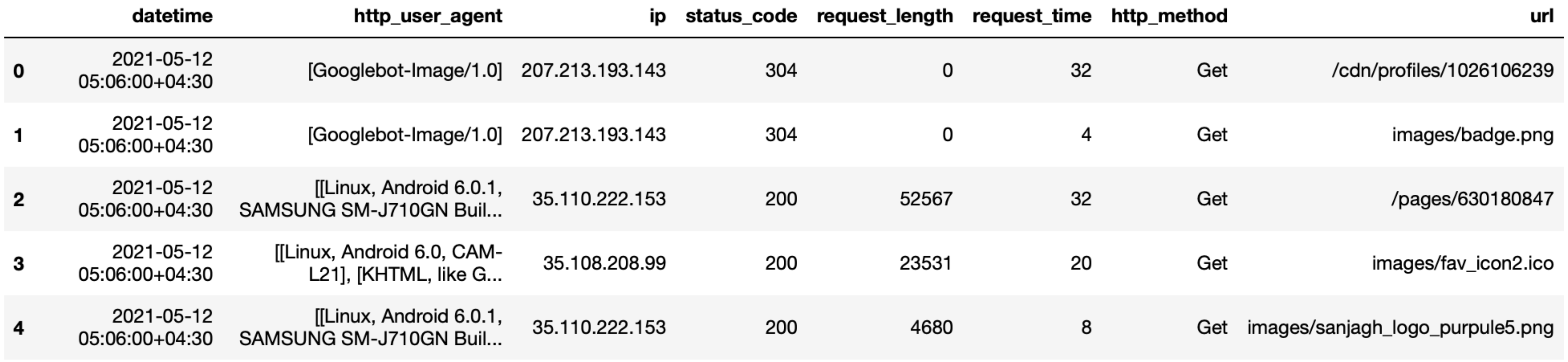
RC – FINAL PROJECT

Arman Aminian

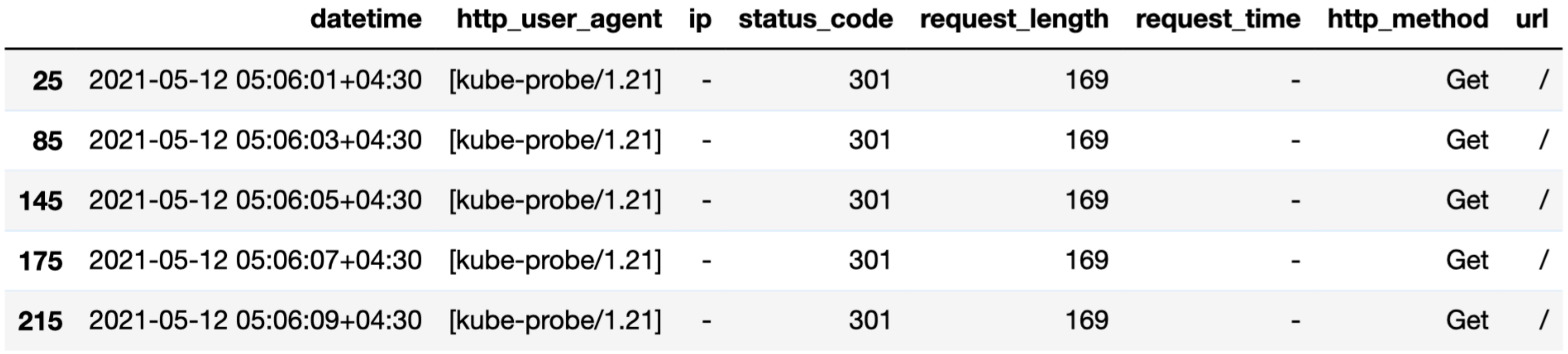
Mentor : **Mobin** Nik khesal

Visualize - Preview



	datetime	http_user_agent	ip	status_code	request_length	request_time	http_method	url
0	2021-05-12 05:06:00+04:30	[Googlebot-Image/1.0]	207.213.193.143	304	0	32	Get	/cdn/profiles/1026106239
1	2021-05-12 05:06:00+04:30	[Googlebot-Image/1.0]	207.213.193.143	304	0	4	Get	images/badge.png
2	2021-05-12 05:06:00+04:30	[[Linux, Android 6.0.1, SAMSUNG SM-J710GN Buil...	35.110.222.153	200	52567	32	Get	/pages/630180847
3	2021-05-12 05:06:00+04:30	[[Linux, Android 6.0, CAM-L21], [KHTML, like G...	35.108.208.99	200	23531	20	Get	images/fav_icon2.ico
4	2021-05-12 05:06:00+04:30	[[Linux, Android 6.0.1, SAMSUNG SM-J710GN Buil...	35.110.222.153	200	4680	8	Get	images/sanjagh_logo_purpule5.png

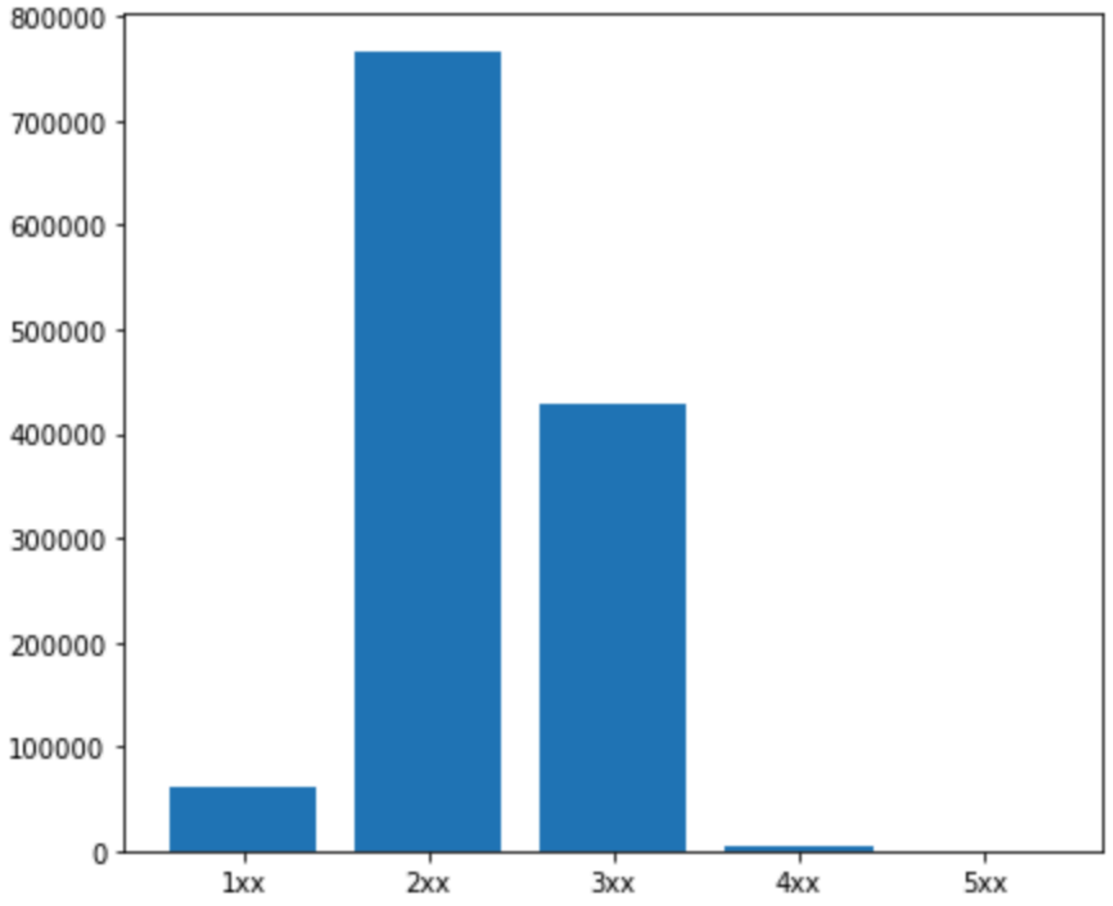
Visualize – IP : User-Agent

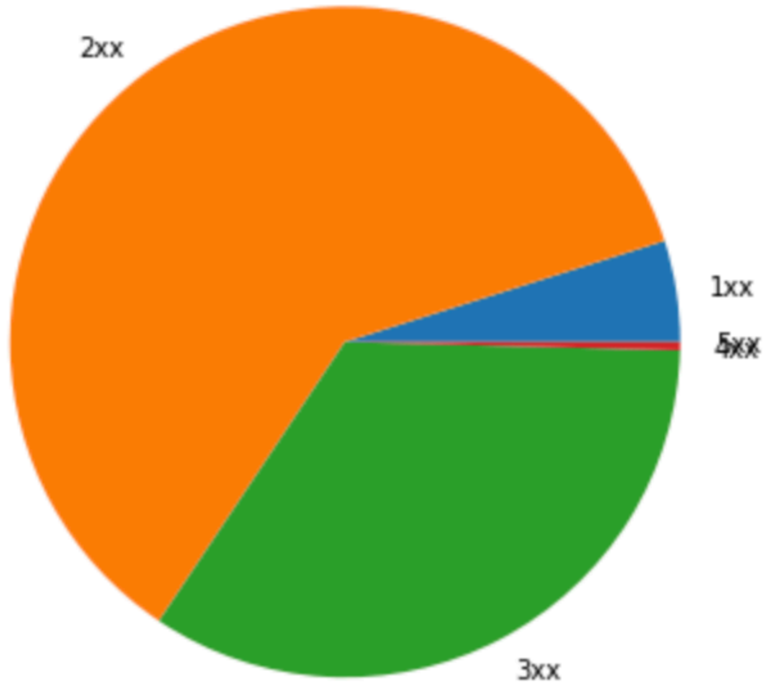


	datetime	http_user_agent	ip	status_code	request_length	request_time	http_method	url
25	2021-05-12 05:06:01+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
85	2021-05-12 05:06:03+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
145	2021-05-12 05:06:05+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
175	2021-05-12 05:06:07+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
215	2021-05-12 05:06:09+04:30	[kube-probe/1.21]	-	301	169	-	Get	/

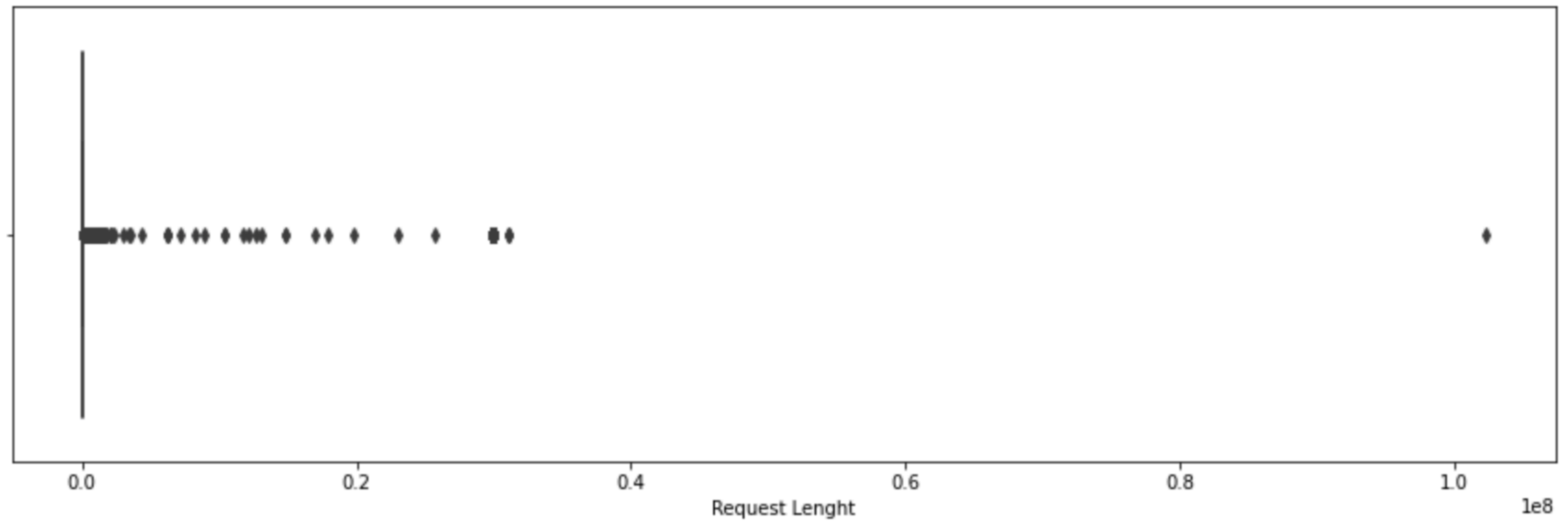
```
len(train[train.ip == '-'] == train[[s == ['kube-probe/1.21'] for s in train.http_user_agent]]) = len(train[train.ip == '-'])
```

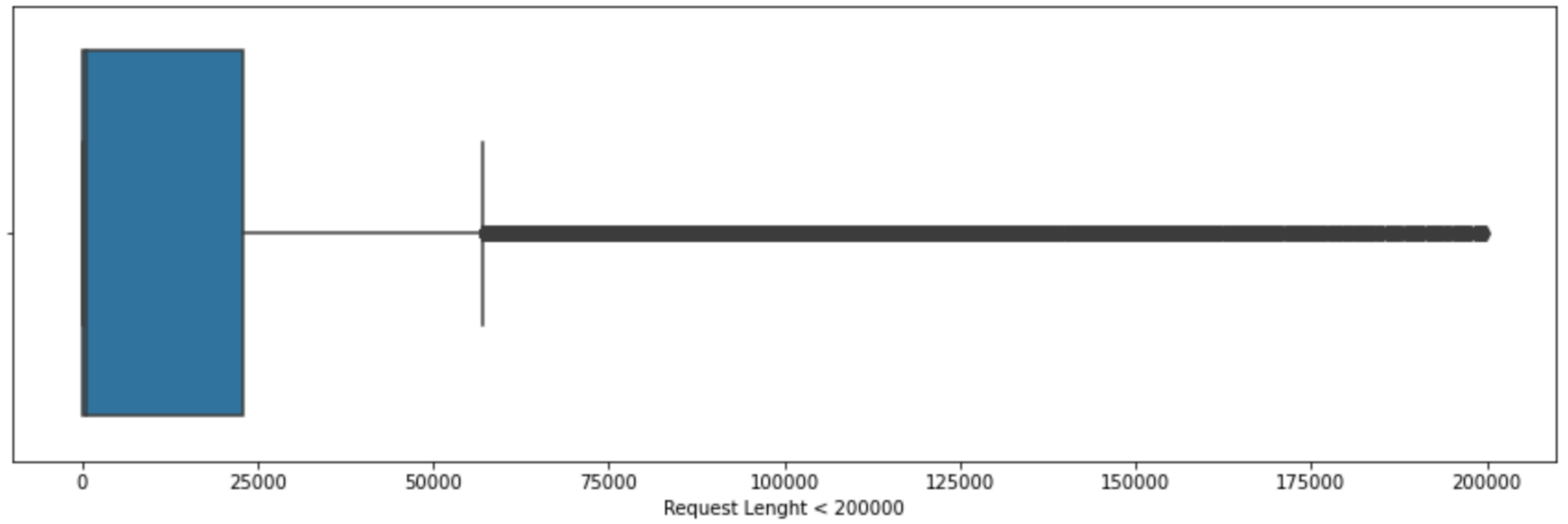
Visualize – Status Code



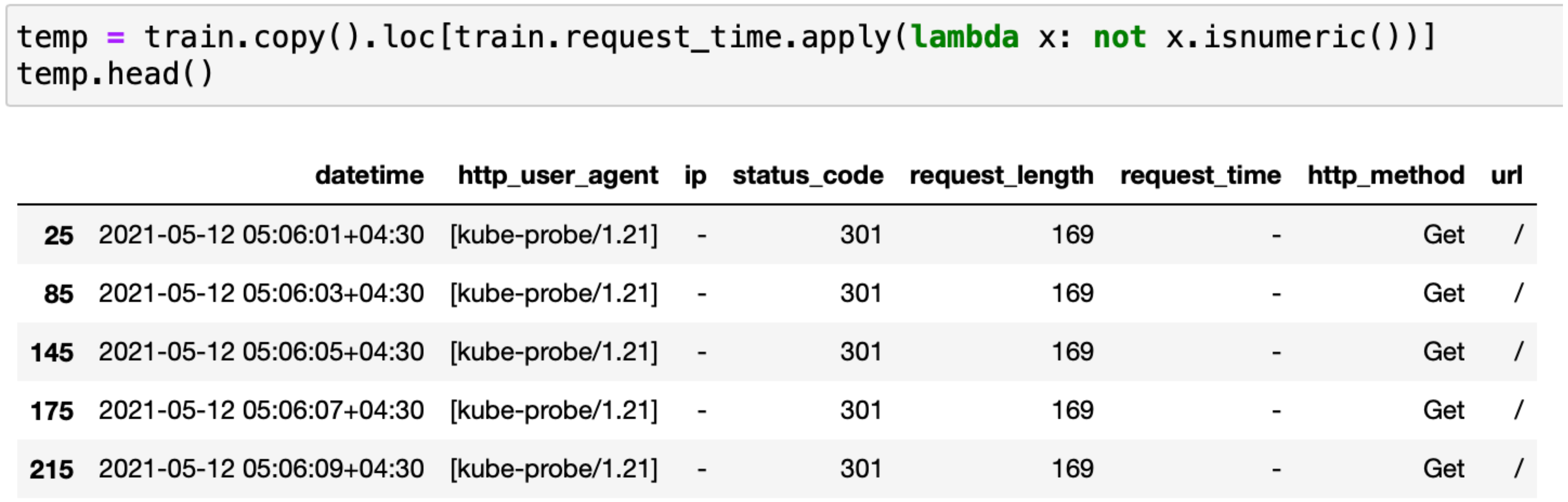


Visualize – Request Length





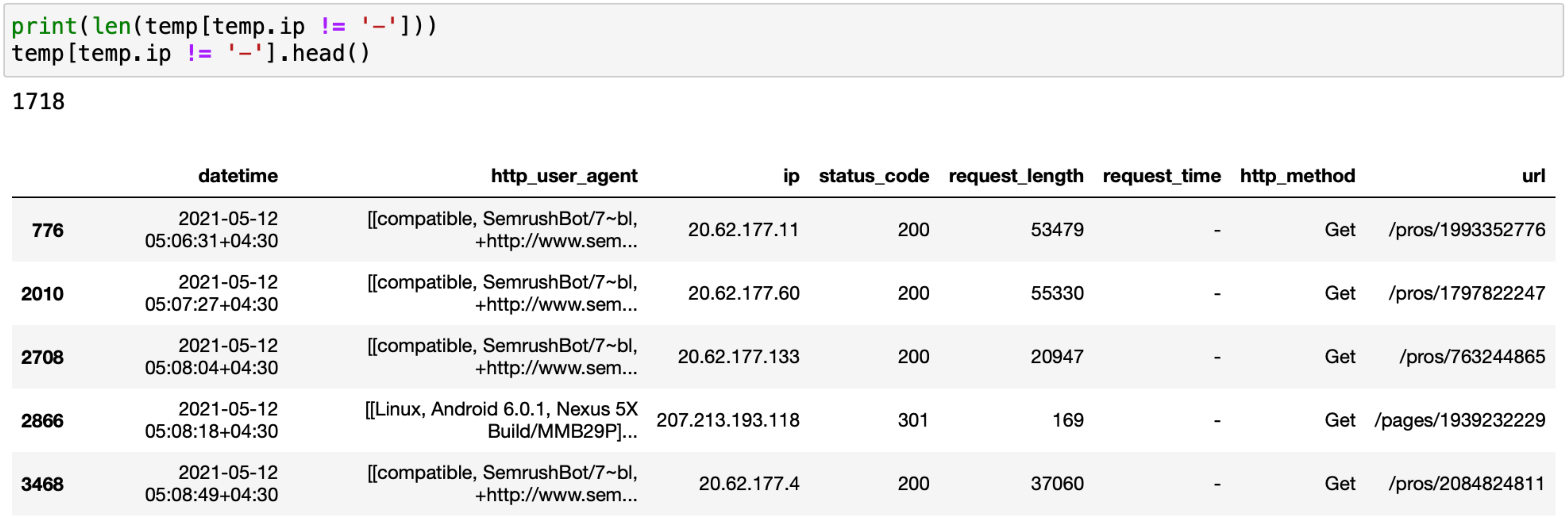
Visualize – Request Time (1)



```python
temp = train.copy().loc[train.request_time.apply(lambda x: not x.isnumeric())]
temp.head()
```

	datetime	http_user_agent	ip	status_code	request_length	request_time	http_method	url
25	2021-05-12 05:06:01+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
85	2021-05-12 05:06:03+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
145	2021-05-12 05:06:05+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
175	2021-05-12 05:06:07+04:30	[kube-probe/1.21]	-	301	169	-	Get	/
215	2021-05-12 05:06:09+04:30	[kube-probe/1.21]	-	301	169	-	Get	/

Visualize – Request Time (2)

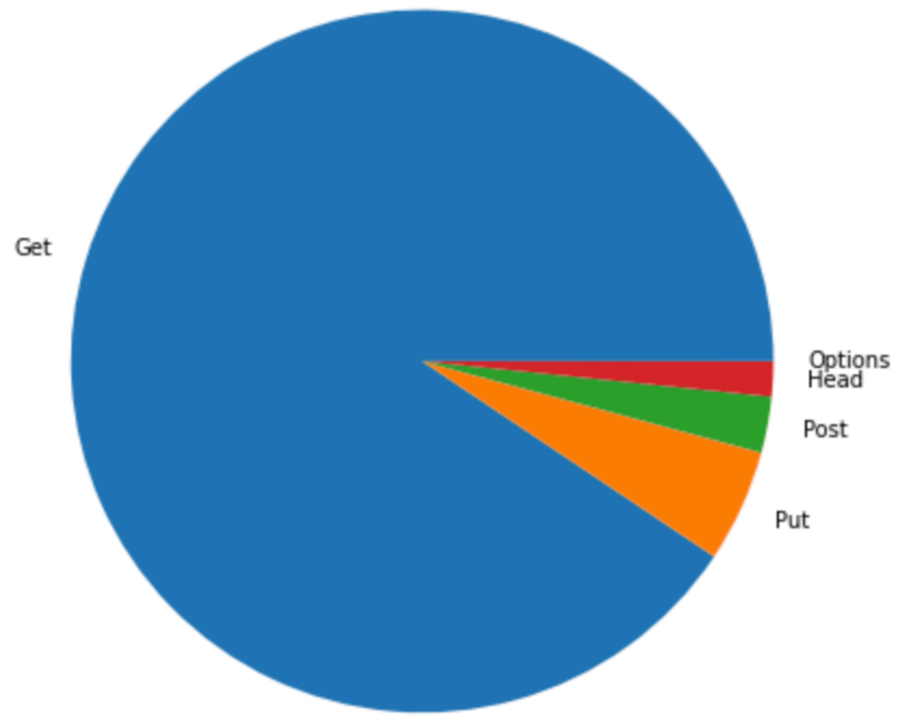


```python
print(len(temp[temp.ip != '-']))
temp[temp.ip != '-'].head()
```

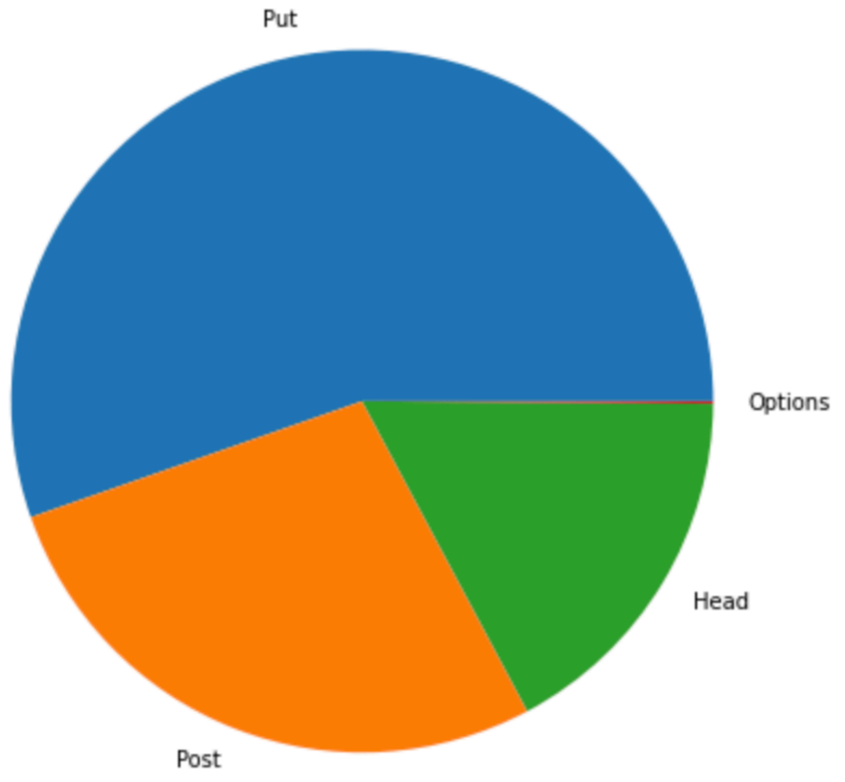
```
1718
```

	datetime	http_user_agent	ip	status_code	request_length	request_time	http_method	url
776	2021-05-12 05:06:31+04:30	[[compatible, SemrushBot/7~bl, +http://www.sem...	20.62.177.11	200	53479	-	Get	/pros/1993352776
2010	2021-05-12 05:07:27+04:30	[[compatible, SemrushBot/7~bl, +http://www.sem...	20.62.177.60	200	55330	-	Get	/pros/1797822247
2708	2021-05-12 05:08:04+04:30	[[compatible, SemrushBot/7~bl, +http://www.sem...	20.62.177.133	200	20947	-	Get	/pros/763244865
2866	2021-05-12 05:08:18+04:30	[[Linux, Android 6.0.1, Nexus 5X Build/MMB29P]...	207.213.193.118	301	169	-	Get	/pages/1939232229
3468	2021-05-12 05:08:49+04:30	[[compatible, SemrushBot/7~bl, +http://www.sem...	20.62.177.4	200	37060	-	Get	/pros/2084824811

Visualize – HTTP Method (1)

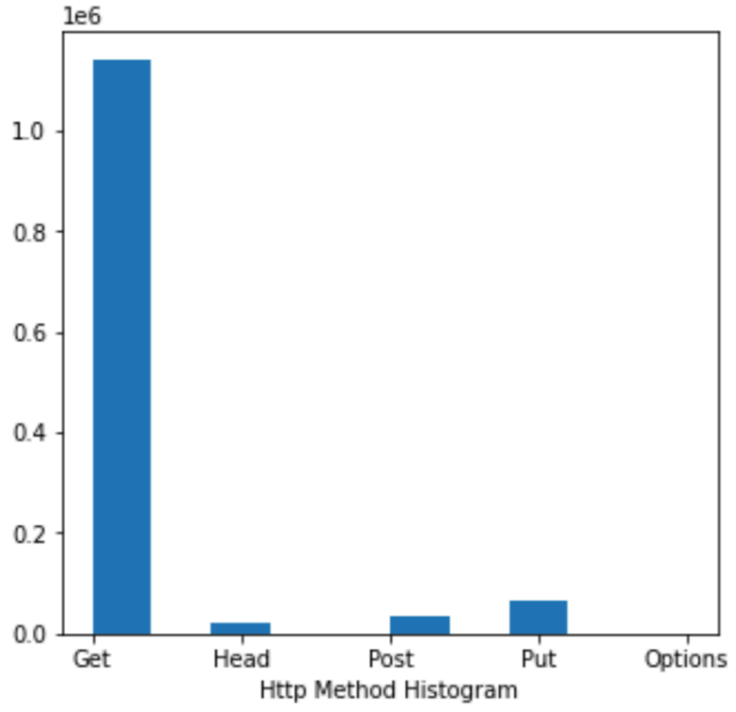


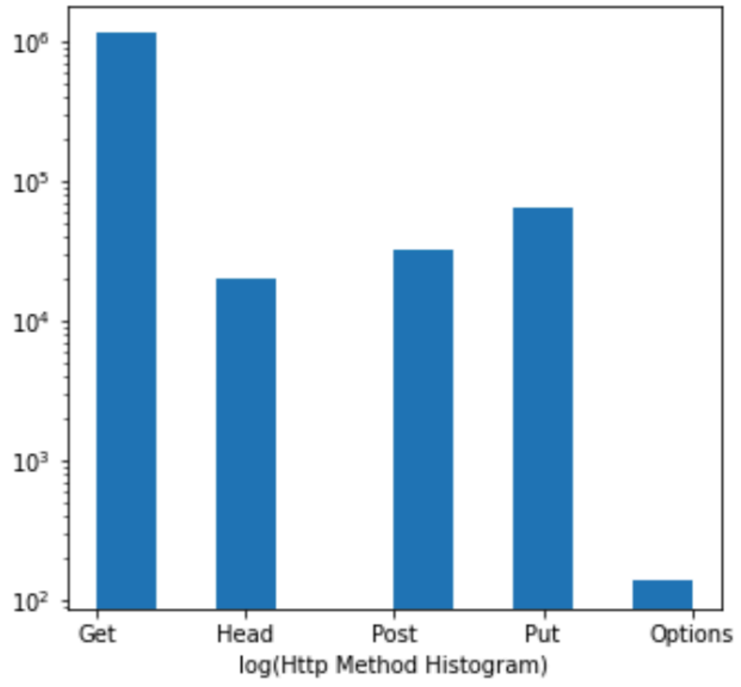
Http Method Pie Plot

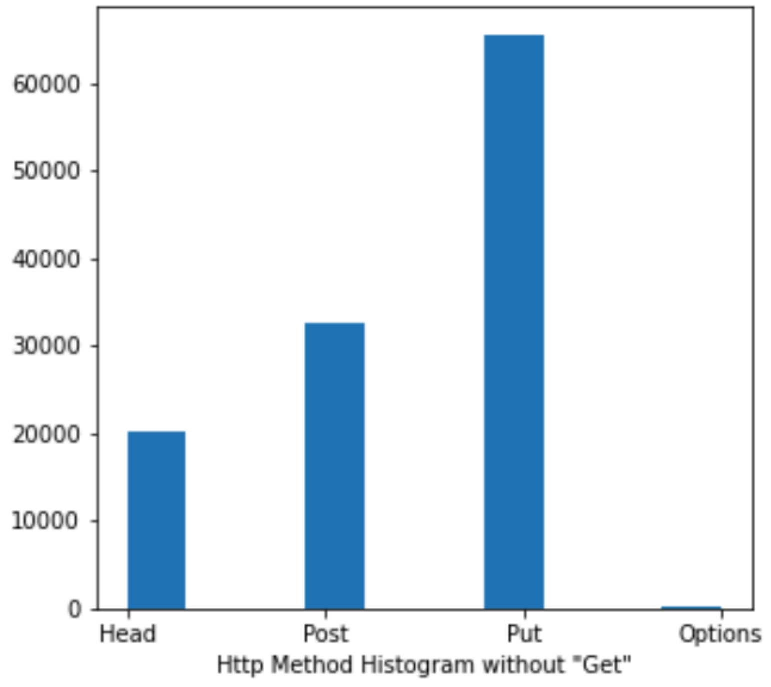


Http Method Pie Plot without "Get"

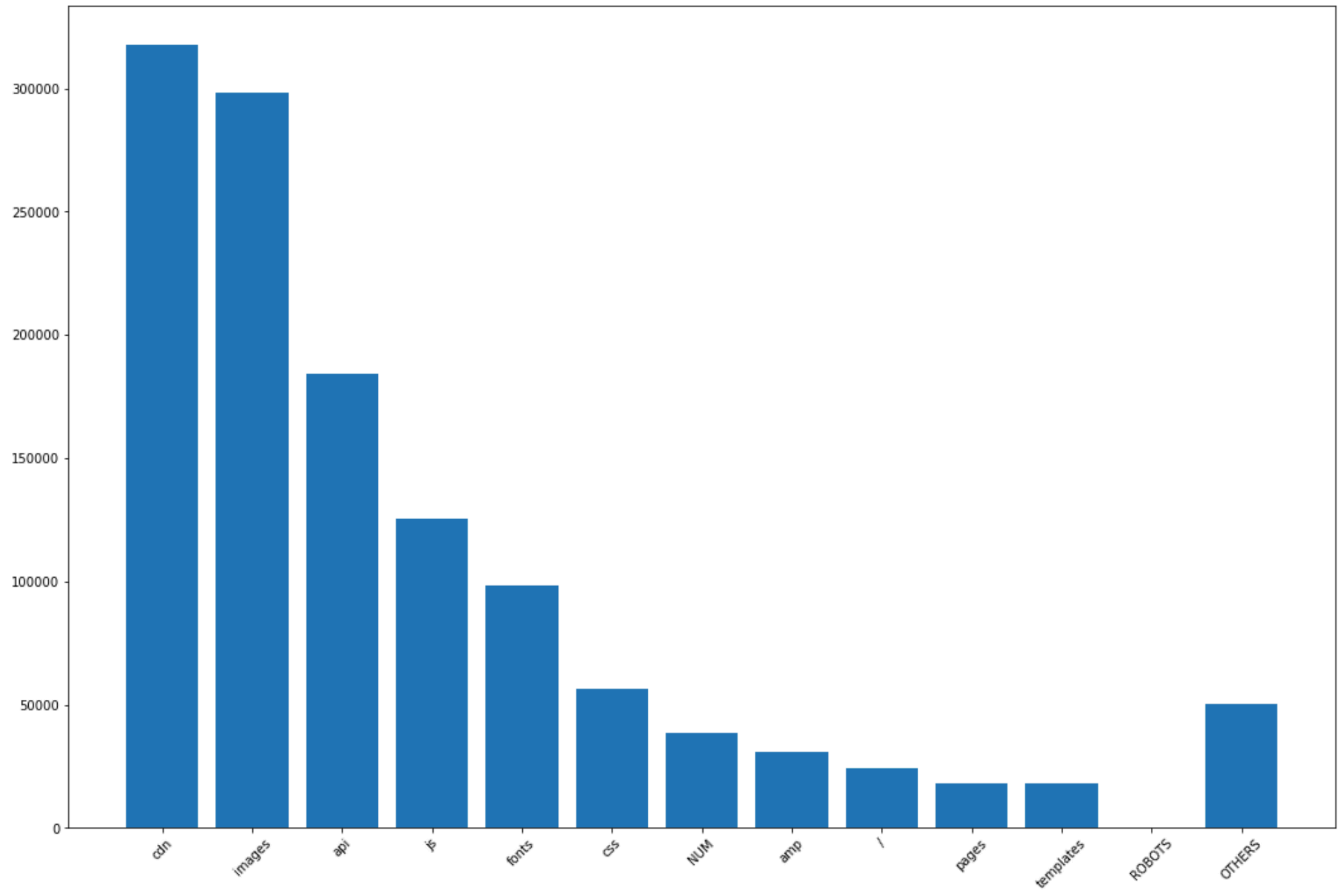
Visualize – HTTP Method (2)



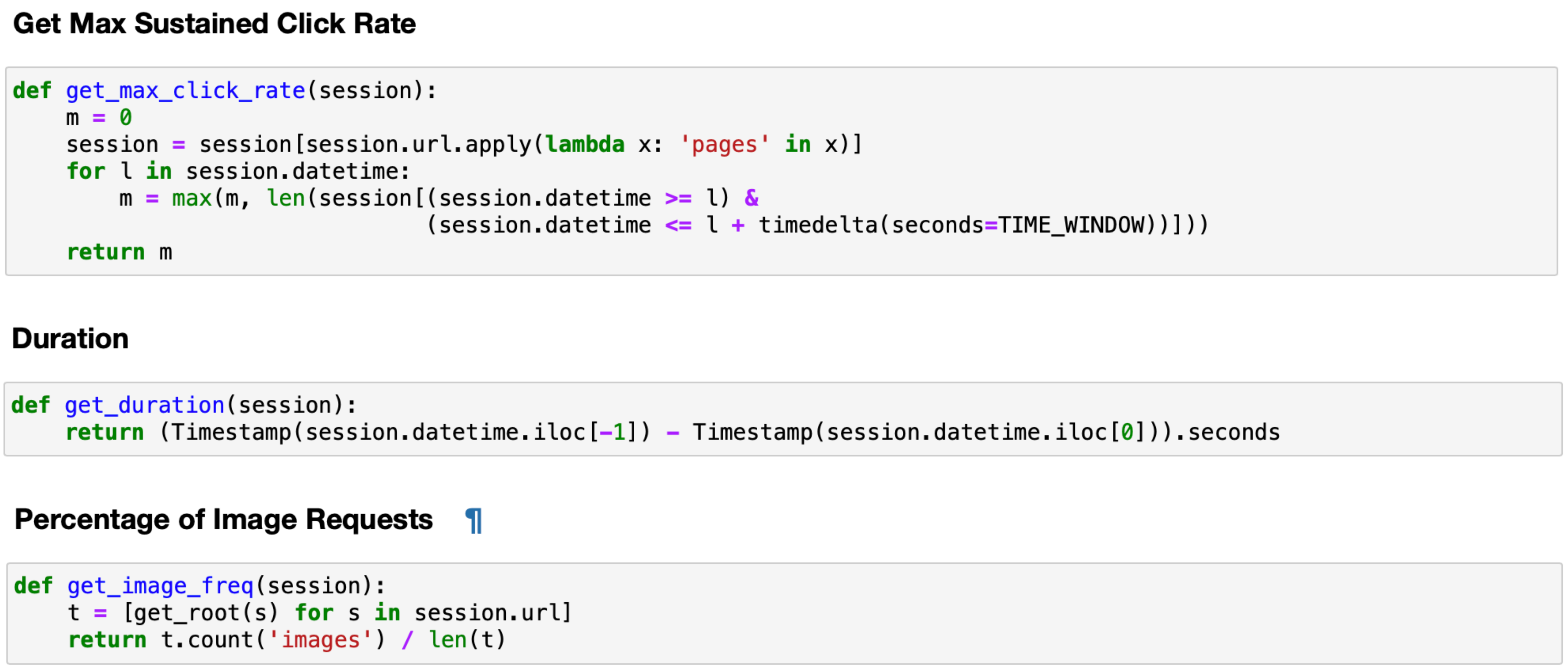




Visualize - URL



Feature Extraction – Session Identification (1)



Get Max Sustained Click Rate

```python
def get_max_click_rate(session):
    m = 0
    session = session[session.url.apply(lambda x: 'pages' in x)]
    for l in session.datetime:
        m = max(m, len(session[(session.datetime >= l) &
                               (session.datetime <= l + timedelta(seconds=TIME_WINDOW))]))
    return m
```

Duration

```python
def get_duration(session):
    return (Timestamp(session.datetime.iloc[-1]) - Timestamp(session.datetime.iloc[0])).seconds
```

Percentage of Image Requests ¶

```python
def get_image_freq(session):
    t = [get_root(s) for s in session.url]
    return t.count('images') / len(t)
```

Feature Extraction – Session Identification (2)

Percentage of 4xx Error Responses

```
def get_4xx_freq(session):
    status_counts = get_categorical_status_code_counts(session.status_code)
    return status_counts[3] / len(session)
```

Percentage of Page Requests

```
def get_page_freq(session):
    t = [get_root(s) for s in session.url]
    return (t.count('pages') + t.count('php') + t.count('asp')) / len(t)
```

Percentage of Head Requests

```
def get_head_freq(session):
    return len(session[session.http_method == 'Head']) / len(session)
```

Feature Extraction – Session Identification (3)

Robots Request

```python
def has_robots_req(session):
    t = ['robots' in s for s in session.url]
    return int(sum(t) > 0)
```

Is Bot

```python
def is_bot(session):
    user_agent = parse(session.http_user_agent.iloc[0])
    return int(user_agent.is_bot)
```

Preprocess

- head_freq
- req_num
- img_freq
- page_freq
- status_4xx_freq
- max_click_rate
- has_robots
- duration
- is_bot
- req_freq

MinMaxScaler

Approximate Labeling

```python
def estimate_label(x):
    h = int(x.duration > DURATION_THRESHOLD) + int((x.page_freq / (x.img_freq + EPSILON)) >= 10)
    return int(x.is_bot or x.has_robots or h >= 2)
```

Manual Labeling

Base Model

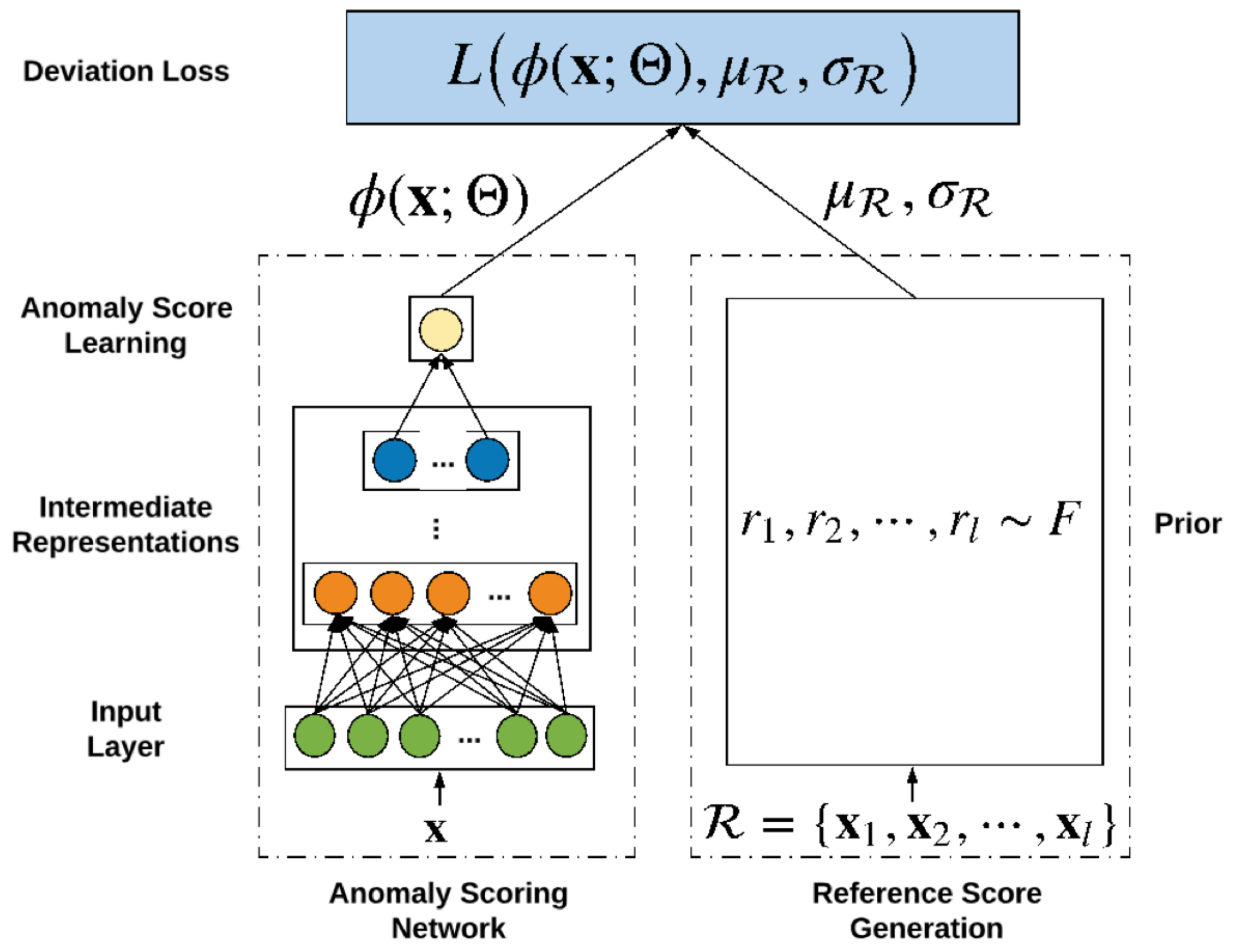
- PCA Based Model

  Precision: 0.32013

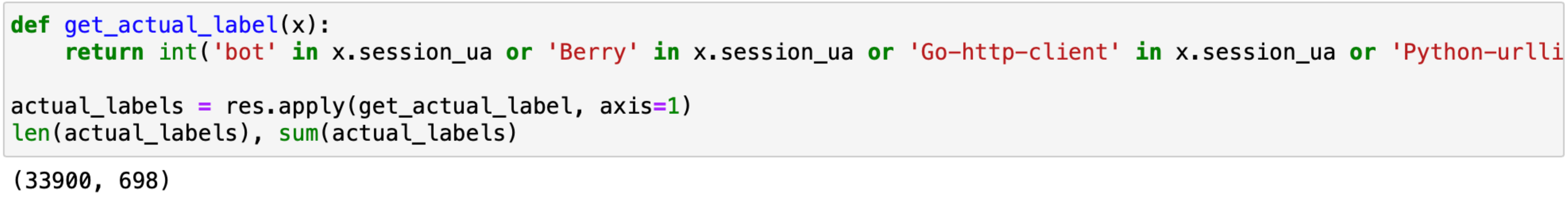
Semi Supervised

- labeling data is usually time-consuming and costly
- very few labeled instances

Deep Anomaly Detection with Deviation Networks (Dev-Net)


Deviation Loss
$L\big(\phi(\mathbf{x};\Theta), \mu_{\mathcal{R}}, \sigma_{\mathcal{R}}\big)$
$\phi(\mathbf{x};\Theta)$
$\mu_{\mathcal{R}}, \sigma_{\mathcal{R}}$
Anomaly Score Learning
Intermediate Representations
Input Layer
$\mathbf{x}$
$r_1, r_2, \cdots, r_l \sim F$
Prior
$\mathcal{R} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_l\}$
Anomaly Scoring Network
Reference Score Generation

Actual Labels



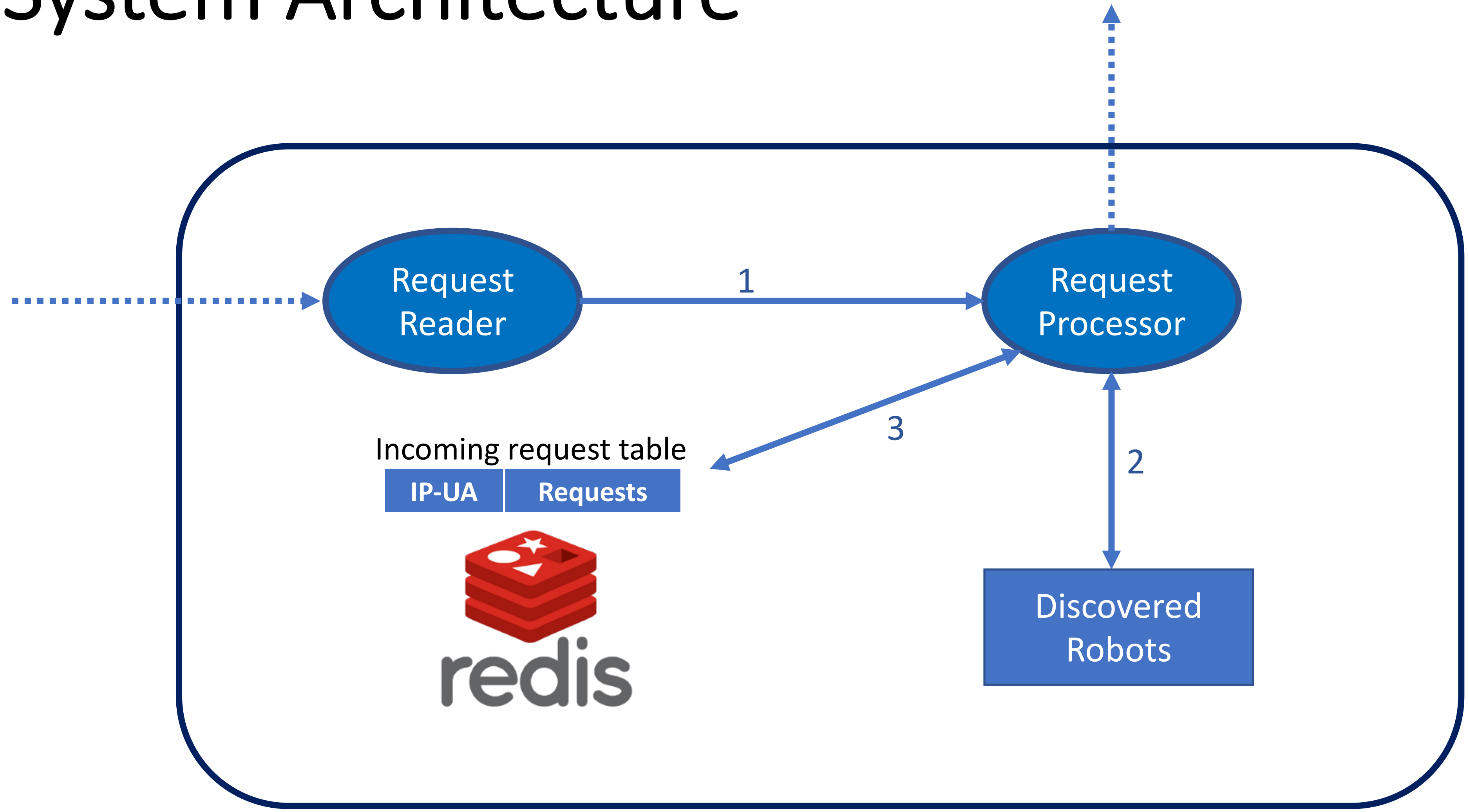
```
def get_actual_label(x):
    return int('bot' in x.session_ua or 'Berry' in x.session_ua or 'Go-http-client' in x.session_ua or 'Python-urlli

actual_labels = res.apply(get_actual_label, axis=1)
len(actual_labels), sum(actual_labels)
```

```
(33900, 698)
```

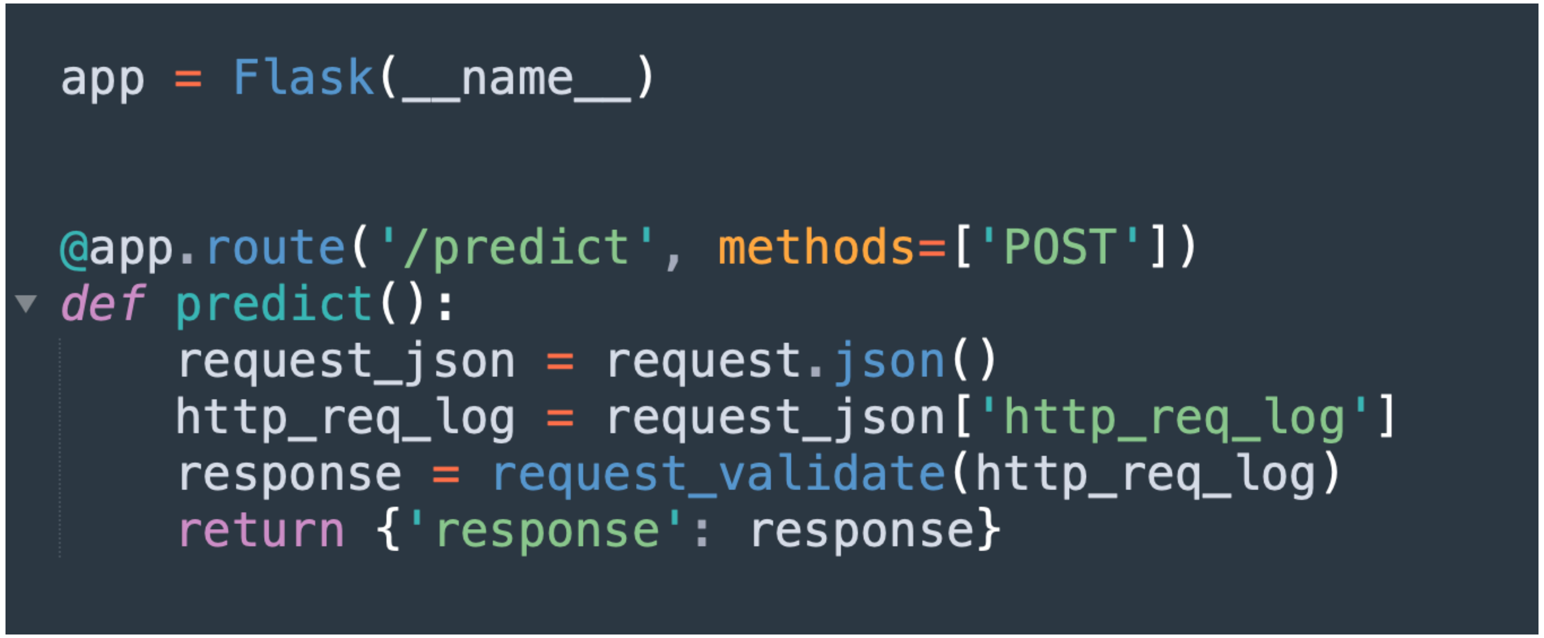
Evaluation

• Precision:	0.9003
• Recall:	0.8281
• Accuracy:	0.9946

System Architecture



API - Flask



```
app = Flask(__name__)


@app.route('/predict', methods=['POST'])
def predict():
    request_json = request.json()
    http_req_log = request_json['http_req_log']
    response = request_validate(http_req_log)
    return {'response': response}
```

References

- Deep Anomaly Detection with Deviation Networks
  - Guansong Pang , Chunhua Shen , Anton van den Hengel
- Web robot detection: A probabilistic reasoning approach
  - Athena Stassopoulou ,Marios D. Dikaiakos
- Real-time web crawler detection
  - Andoena Balla ,Athena Stassopoulou ,Marios D. Dikaiakos
- Web Robot Detection in Academic Publishing
  - Athanasios Lagopoulos , Grigorios Tsoumakas , Georgios Papadopoulos

GitHub Repository

- github.com/arman-aminian/network-anomaly-detection

Special thanks to

- **Mobin** Nik khesal
- **Azin** Azarkar
- **Yasin** Orouskhani
- **You** for listening.