mini2

May 11, 2024

[2]:

```
import os
import xgboost as xgb
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader, TensorDataset
from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier

from tensorflow.keras import Input
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential



from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, f1_score, precision_score,␣
 ↪recall_score


data = pd.read_csv(r'C:\Users\BRINDHA\Desktop\web-page-phishing.csv')
data
```

[2]:

```
        url_length  n_dots  n_hypens  n_underline  n_slash  n_questionmark  \
0               37       3         0            0        0               0
1               77       1         0            0        0               0
2              126       4         1            2        0               1
```

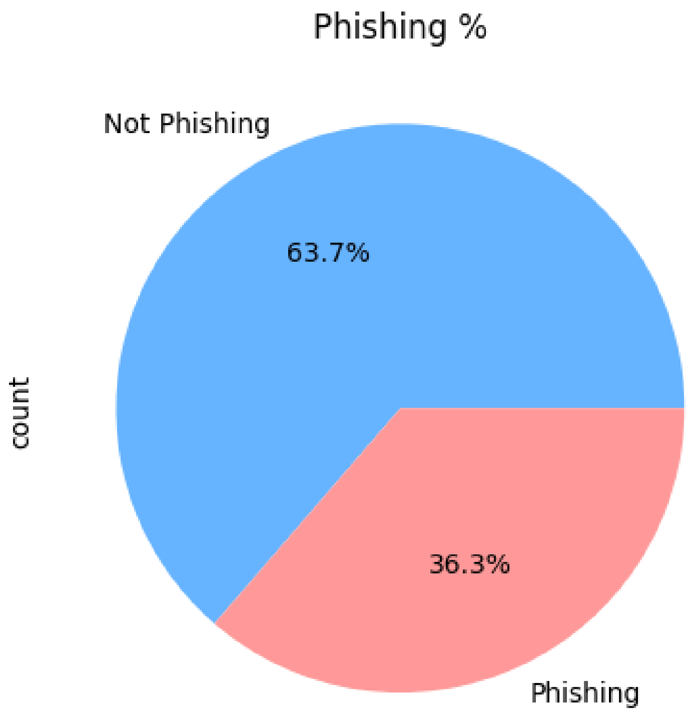
```
3                18      2        0            0       0               0
4                55      2        2            0       0               0
…              …      …        …           …      …             …
100072           23      3        1            0       0               0
100073           34      2        0            0       0               0
100074           70      2        1            0       5               0
100075           28      2        0            0       1               0
100076           16      2        0            0       0               0

        n_equal  n_at  n_and  n_exclamation  n_space  n_tilde  n_comma  \
0             0     0      0              0        0        0        0
1             0     0      0              0        0        0        0
2             3     0      2              0        0        0        0
3             0     0      0              0        0        0        0
4             0     0      0              0        0        0        0
…           …   …    …            …      …      …      …
100072        0     0      0              0        0        0        0
100073        0     0      0              0        0        0        0
100074        0     0      0              0        0        0        0
100075        0     0      0              0        0        0        0
100076        0     0      0              0        0        0        0

        n_plus  n_asterisk  n_hastag  n_dollar  n_percent  n_redirection  \
0            0           0         0         0          0              0
1            0           0         0         0          0              1
2            0           0         0         0          0              1
3            0           0         0         0          0              1
4            0           0         0         0          0              1
…         …        …       …       …        …            …
100072       0           0         0         0          0              0
100073       0           0         0         0          0              2
100074       0           0         0         0          0              0
100075       0           0         0         0          0              0
100076       0           0         0         0          0              0

        phishing
0              0
1              1
2              1
3              0
4              0
…           …
100072         0
100073         0
100074         1
100075         1
100076         0
```

```
[100077 rows x 20 columns]
```

[3]:
```
is_there_null_values =  data.isna().any().any()
is_there_null_values
```

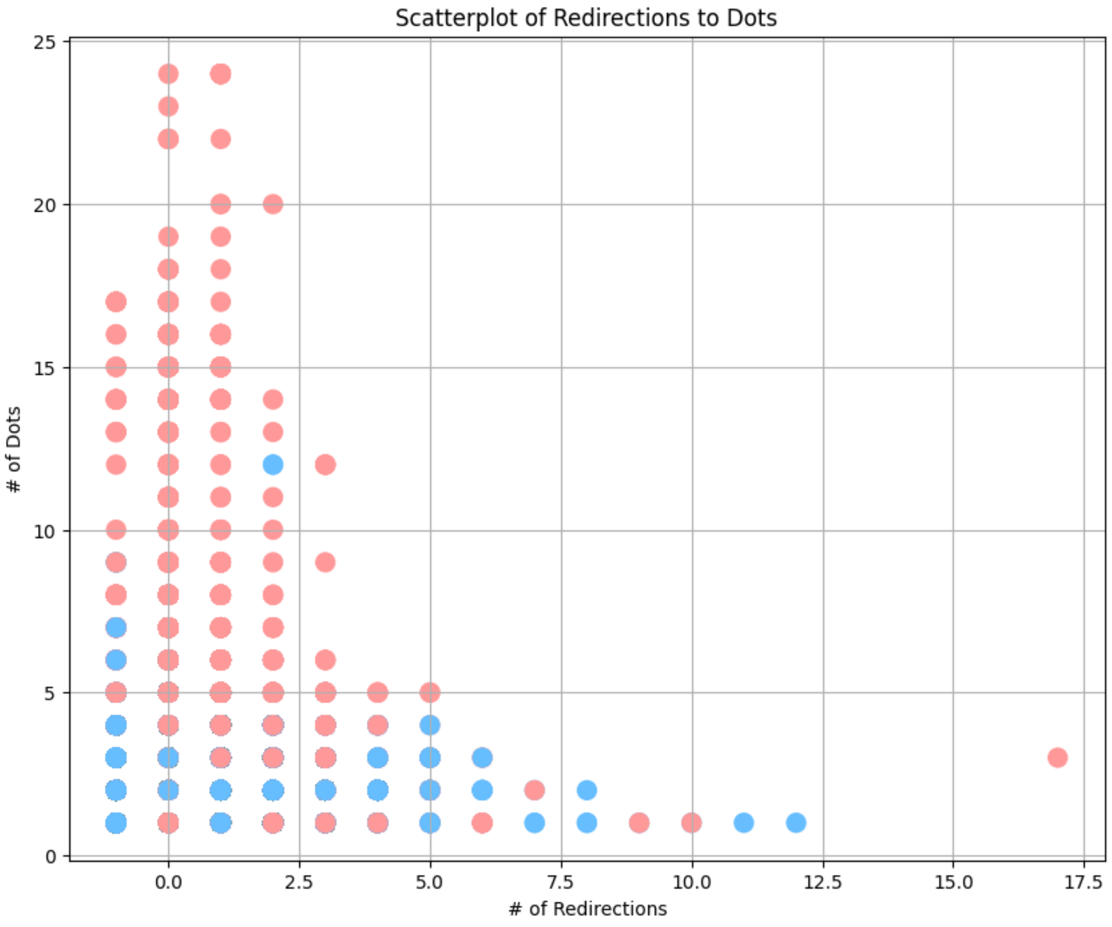
[3]:
```
False
```

[5]:
```
colors = ['#66b3ff', '#ff9999']
data_counts = data['phishing'].value_counts()
data_counts.plot.pie(autopct = '%1.1f%%', labels = ['Not Phishing',␣
  ↪'Phishing'], colors = colors)
plt.title('Phishing %')

plt.savefig('g1.png', bbox_inches = 'tight')
plt.show()
```



[9]:
```
import matplotlib.pyplot as plt

# Create a figure and axis object with a specified size
fig, ax = plt.subplots(figsize=(10, 8))
```

```python
# Define colors for the scatterplot
data_colors = {1: '#ff9999',  # Red for phishing
               0: '#66beff'}  # Blue for non-phishing

# Plot the scatterplot with modified color and size
ax.scatter(data=data, x='n_redirection', y='n_dots', color=data['phishing'].
 ↪map(data_colors), s=100)

# Set labels and title
ax.set_xlabel('# of Redirections')
ax.set_ylabel('# of Dots')
ax.set_title('Scatterplot of Redirections to Dots')

# Add grid
ax.grid(True)

# Show the plot
plt.show()
```



[11]:

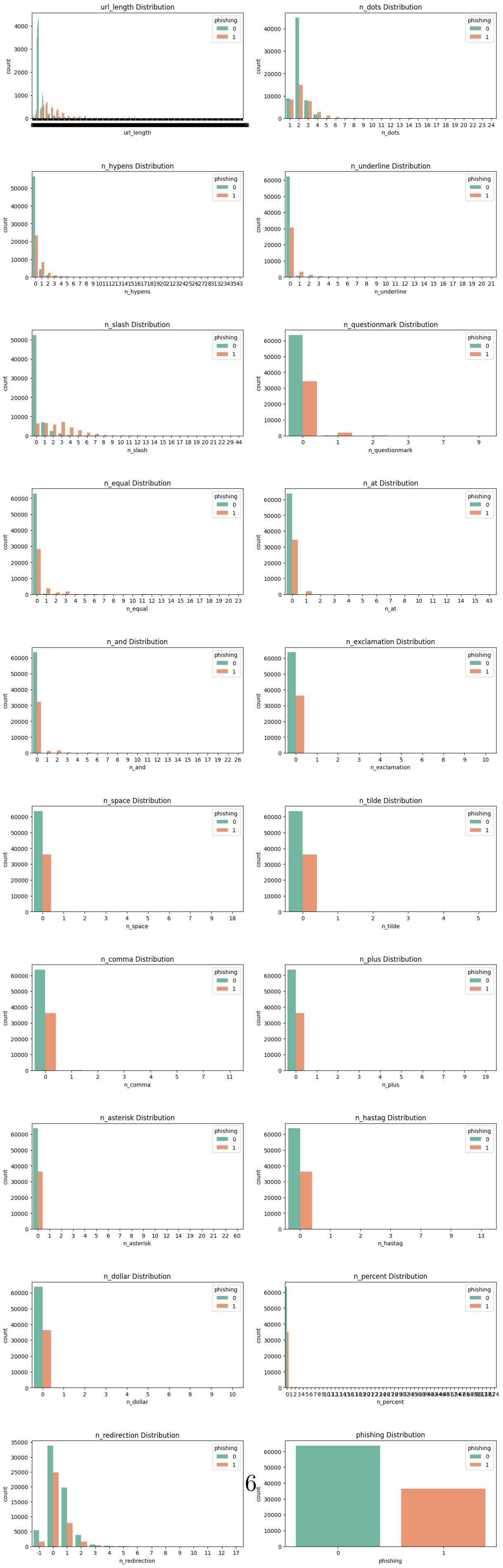
```
target = 'phishing'
data[target].unique()
```

[11]:

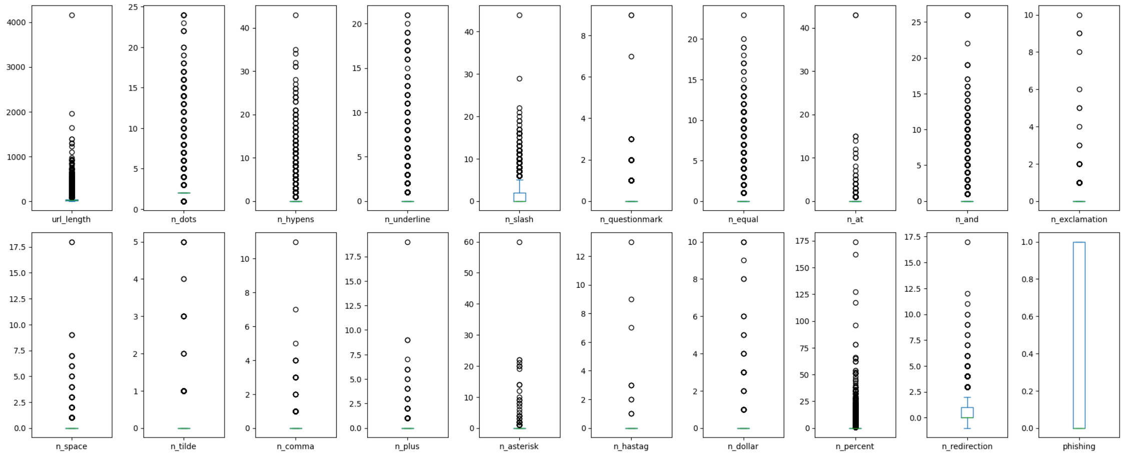
```
array([0, 1], dtype=int64)
```

[20]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

def visualizeData(data: pd.DataFrame, visualizableFields=[], target=None, gap=0.
 ↪5, padding=5):
    n = len(visualizableFields)
    n_rows = n // 2 + (n % 2 != 0)

    fig, axes = plt.subplots(n_rows, 2, figsize=(15, 5*n_rows),␣
 ↪gridspec_kw={'hspace': gap})

    for idx, value in enumerate(visualizableFields):
        ax = axes[idx//2, idx%2]
        sns.countplot(x=value, hue=target, data=data, palette="Set2", ax=ax)
        ax.set_title(f"{value} Distribution")


    plt.show()
visualizeData(data, data.columns, target)
```


url_length Distribution
count
url_length
phishing
0
1
0
1000
2000
3000
4000
n_dots Distribution
n_dots
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 24
0
10000
20000
30000
40000
n_hypens Distribution
n_hypens
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 23 24 25 26 27 28 31 32 34 35 43
0
10000
20000
30000
40000
50000
n_underline Distribution
n_underline
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
0
10000
20000
30000
40000
50000
60000
n_slash Distribution
n_slash
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 29 44
n_questionmark Distribution
n_questionmark
0 1 2 3 7 9
n_equal Distribution
n_equal
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 23
n_at Distribution
n_at
0 1 2 3 4 5 6 7 8 10 11 12 14 15 43
n_and Distribution
n_and
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 19 22 26
n_exclamation Distribution
n_exclamation
0 1 2 3 4 5 6 8 9 10
n_space Distribution
n_space
0 1 2 3 4 5 6 7 9 18
n_tilde Distribution
n_tilde
0 1 2 3 4 5
n_comma Distribution
n_comma
0 1 2 3 4 5 7 11
n_plus Distribution
n_plus
0 1 2 3 4 5 6 7 9 19
n_asterisk Distribution
n_asterisk
0 1 2 3 4 5 6 7 8 9 10 12 14 19 20 21 22 60
n_hastag Distribution
n_hastag
0 1 2 3 7 9 13
n_dollar Distribution
n_dollar
0 1 2 3 4 5 6 8 9 10
n_percent Distribution
n_percent
n_redirection Distribution
n_redirection
-1 0 1 2 3 4 5 6 7 8 9 10 11 12 17
0
5000
10000
15000
20000
25000
30000
35000
phishing Distribution
phishing
0 1

[23]:
```
data.plot(kind='box',subplots=True,layout=(5,10),figsize=(20,20))
plt.xlabel('Column')
plt.ylabel('Value')
plt.title('Box Plots of Each Column')
plt.tight_layout()  # Adjust spacing between subplots

# Display the plot
plt.show()
```



[25]:
```
data_model_n_slash = data.groupby(['n_slash', 'phishing']).size()
data_model_n_slash_phishing_df = data_model_n_slash.reset_index(name='count')
data_model_n_slash_phishing_df
```

[25]:
```
    n_slash  phishing  count
0         0         0  52477
1         0         1   6462
2         1         0   7045
3         1         1   6477
4         2         0   2317
5         2         1   5755
6         3         0   1223
7         3         1   7135
8         4         0    293
9         4         1   4383
10        5         0    225
11        5         1   2774
12        6         0     87
13        6         1   1578
14        7         0     34
```

```
15         7          1    862
16         8          0      9
17         8          1    422
18         9          0      3
19         9          1    228
20        10          0      2
21        10          1    102
22        11          1     60
23        12          1     75
24        13          1     13
25        14          1      6
26        15          1      3
27        16          1     15
28        17          1      4
29        18          1      1
30        19          1      2
31        20          1      1
32        21          1      1
33        22          1      1
34        29          1      1
35        44          1      1
```

[26]:

```
# Group by 'n_slash' and 'phishing', and count occurrences
data_model_n_slash = data.groupby(['n_slash', 'phishing'], observed=False).
 ↪size()

# Reset the index to flatten the DataFrame
data_model_n_slash_phishing_df = data_model_n_slash.reset_index(name='count')


bins = list(range(0, data_model_n_slash_phishing_df['n_slash'].max() + 5, 3)) ␣
 ↪# Create bins of size 4, starting from 0
data_model_n_slash_phishing_df['n_slash_bin'] = pd.
 ↪cut(data_model_n_slash_phishing_df['n_slash'], bins=bins, right=False)

data_model_n_slash_phishing_df
```

[26]:

```
    n_slash  phishing  count n_slash_bin
0         0         0  52477      [0, 3)
1         0         1   6462      [0, 3)
2         1         0   7045      [0, 3)
3         1         1   6477      [0, 3)
4         2         0   2317      [0, 3)
5         2         1   5755      [0, 3)
6         3         0   1223      [3, 6)
7         3         1   7135      [3, 6)
8         4         0    293      [3, 6)
```

```
9         4         1    4383      [3, 6)
10        5         0     225      [3, 6)
11        5         1    2774      [3, 6)
12        6         0      87      [6, 9)
13        6         1    1578      [6, 9)
14        7         0      34      [6, 9)
15        7         1     862      [6, 9)
16        8         0       9      [6, 9)
17        8         1     422      [6, 9)
18        9         0       3     [9, 12)
19        9         1     228     [9, 12)
20       10         0       2     [9, 12)
21       10         1     102     [9, 12)
22       11         1      60     [9, 12)
23       12         1      75    [12, 15)
24       13         1      13    [12, 15)
25       14         1       6    [12, 15)
26       15         1       3    [15, 18)
27       16         1      15    [15, 18)
28       17         1       4    [15, 18)
29       18         1       1    [18, 21)
30       19         1       2    [18, 21)
31       20         1       1    [18, 21)
32       21         1       1    [21, 24)
33       22         1       1    [21, 24)
34       29         1       1    [27, 30)
35       44         1       1    [42, 45)
```

[27]:

```
non_phishing_df
```

[27]:

```
   n_slash_bin  n_slash  phishing  count
0       [0, 3)        3         0  61839
1       [3, 6)       12         0   1741
2       [6, 9)       21         0    130
3      [9, 12)       19         0      5
4     [12, 15)        0         0      0
5     [15, 18)        0         0      0
6     [18, 21)        0         0      0
7     [21, 24)        0         0      0
8     [24, 27)        0         0      0
9     [27, 30)        0         0      0
10    [30, 33)        0         0      0
11    [33, 36)        0         0      0
12    [36, 39)        0         0      0
13    [39, 42)        0         0      0
14    [42, 45)        0         0      0
15    [45, 48)        0         0      0
```

[28]:

```
phishing_df.shape == non_phishing_df.shape
```

[28]:

```
True
```

[29]:

```
import pandas as pd

# Define the columns using dictionaries
columns = {
    'n_slash_bin': non_phishing_df['n_slash_bin'],
    'non_phishing_count': non_phishing_df['count'],
    'phishing_count': phishing_df['count'],
    'total_count': phishing_df['count'] + non_phishing_df['count'],
    'phishing_percent': phishing_df['count'] * 100 / (phishing_df['count'] +␣
 ↪non_phishing_df['count']),
    'non_phishing_percent': non_phishing_df['count'] * 100 /␣
 ↪(phishing_df['count'] + non_phishing_df['count'])
}

# Create a DataFrame from the columns
data_model_n_slash_phishing_pivot = pd.DataFrame(columns)

# Display the DataFrame
print(data_model_n_slash_phishing_pivot)
```

```
   n_slash_bin  non_phishing_count  phishing_count  total_count  \
0       [0, 3)               61839           18694        80533
1       [3, 6)                1741           14292        16033
2       [6, 9)                 130            2862         2992
3      [9, 12)                   5             390          395
4     [12, 15)                   0              94           94
5     [15, 18)                   0              22           22
6     [18, 21)                   0               4            4
7     [21, 24)                   0               2            2
8     [24, 27)                   0               0            0
9     [27, 30)                   0               1            1
10    [30, 33)                   0               0            0
11    [33, 36)                   0               0            0
12    [36, 39)                   0               0            0
13    [39, 42)                   0               0            0
14    [42, 45)                   0               1            1
15    [45, 48)                   0               0            0

    phishing_percent  non_phishing_percent
0          23.212844             76.787156
1          89.141146             10.858854
2          95.655080              4.344920
3          98.734177              1.265823
```

```
4     100.000000          0.000000
5     100.000000          0.000000
6     100.000000          0.000000
7     100.000000          0.000000
8            NaN               NaN
9     100.000000          0.000000
10           NaN               NaN
11           NaN               NaN
12           NaN               NaN
13           NaN               NaN
14    100.000000          0.000000
15           NaN               NaN
```

[32]:
```
columns = {
    'n_slash_bin': non_phishing_df['n_slash_bin'],
    'non_phishing_count': non_phishing_df['count'],
    'phishing_count' : phishing_df['count'],
    'total_count' : phishing_df['count'] + non_phishing_df['count'],
    'phishing_percent' : phishing_df['count']* 100 / (phishing_df['count'] +␣
  ↪non_phishing_df['count']),
    'non_phishing_percent' : 100*(1 - (phishing_df['count'] /␣
  ↪(phishing_df['count'] + non_phishing_df['count'])))
}
data_model_n_slash_phishing_pivot = pd.DataFrame(columns)
data_model_n_slash_phishing_pivot['phishing_percent'] =␣
  ↪data_model_n_slash_phishing_pivot['phishing_percent'].fillna(0)
data_model_n_slash_phishing_pivot['non_phishing_percent'] =␣
  ↪data_model_n_slash_phishing_pivot['non_phishing_percent'].fillna(0)
data_model_n_slash_phishing_pivot
```

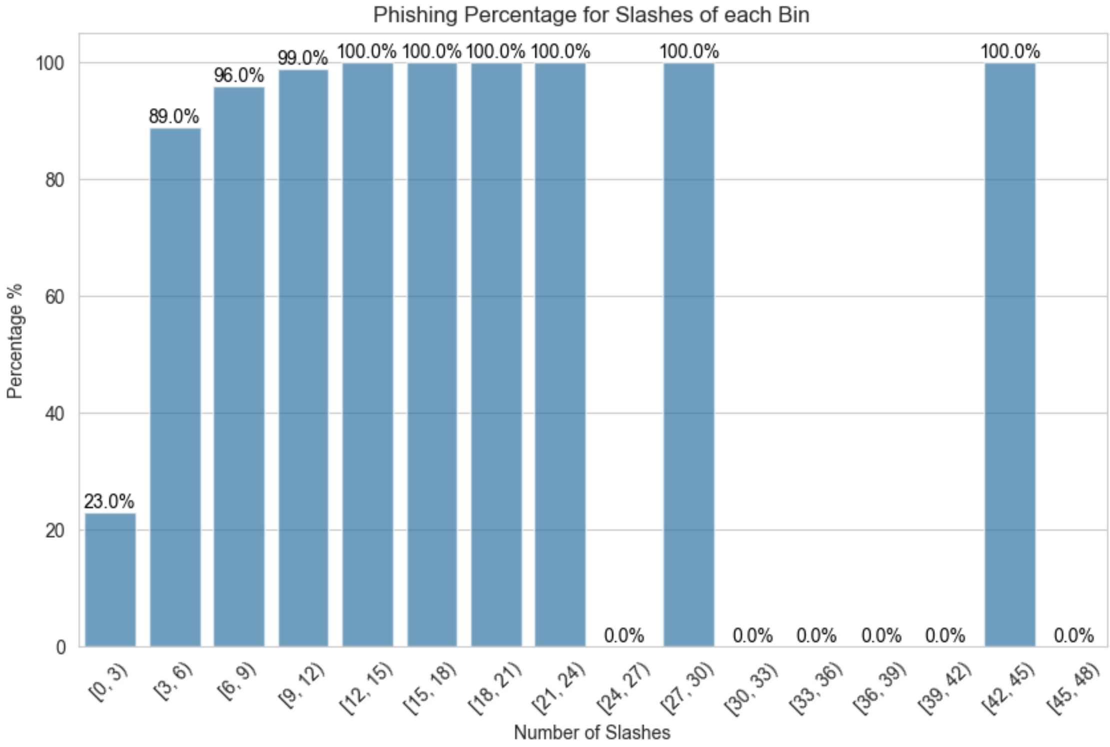
[32]:
```
   n_slash_bin  non_phishing_count  phishing_count  total_count  \
0       [0, 3)               61839           18694        80533
1       [3, 6)                1741           14292        16033
2       [6, 9)                 130            2862         2992
3      [9, 12)                   5             390          395
4     [12, 15)                   0              94           94
5     [15, 18)                   0              22           22
6     [18, 21)                   0               4            4
7     [21, 24)                   0               2            2
8     [24, 27)                   0               0            0
9     [27, 30)                   0               1            1
10    [30, 33)                   0               0            0
11    [33, 36)                   0               0            0
12    [36, 39)                   0               0            0
13    [39, 42)                   0               0            0
14    [42, 45)                   0               1            1
15    [45, 48)                   0               0            0
```

```
    phishing_percent  non_phishing_percent
0          23.212844             76.787156
1          89.141146             10.858854
2          95.655080              4.344920
3          98.734177              1.265823
4         100.000000              0.000000
5         100.000000              0.000000
6         100.000000              0.000000
7         100.000000              0.000000
8           0.000000              0.000000
9         100.000000              0.000000
10          0.000000              0.000000
11          0.000000              0.000000
12          0.000000              0.000000
13          0.000000              0.000000
14        100.000000              0.000000
15          0.000000              0.000000
```

[34]:

```python
sns.set_style("whitegrid")

# Set the width of the bars
bar_width = 0.8  # Adjust this value to increase or decrease the width of the␣
 ↪bars

# Round phishing percentages to the nearest integer
data_model_n_slash_phishing_pivot['rounded_phishing_percent'] =␣
 ↪data_model_n_slash_phishing_pivot['phishing_percent'].round().astype(int)

# Create the bar chart
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='n_slash_bin', y='rounded_phishing_percent',␣
 ↪data=data_model_n_slash_phishing_pivot, errorbar=None, alpha=0.7,␣
 ↪width=bar_width)

for p in ax.patches:
    ax.annotate(f'{p.get_height()}%', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='center', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                textcoords='offset points')

# Set labels and title
plt.xlabel('Number of Slashes')
plt.ylabel('Percentage %')
plt.title('Phishing Percentage for Slashes of each Bin')
```

```
# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

#plt.show()

plt.savefig("phishing_percentage_for_slashes.jpeg")
```

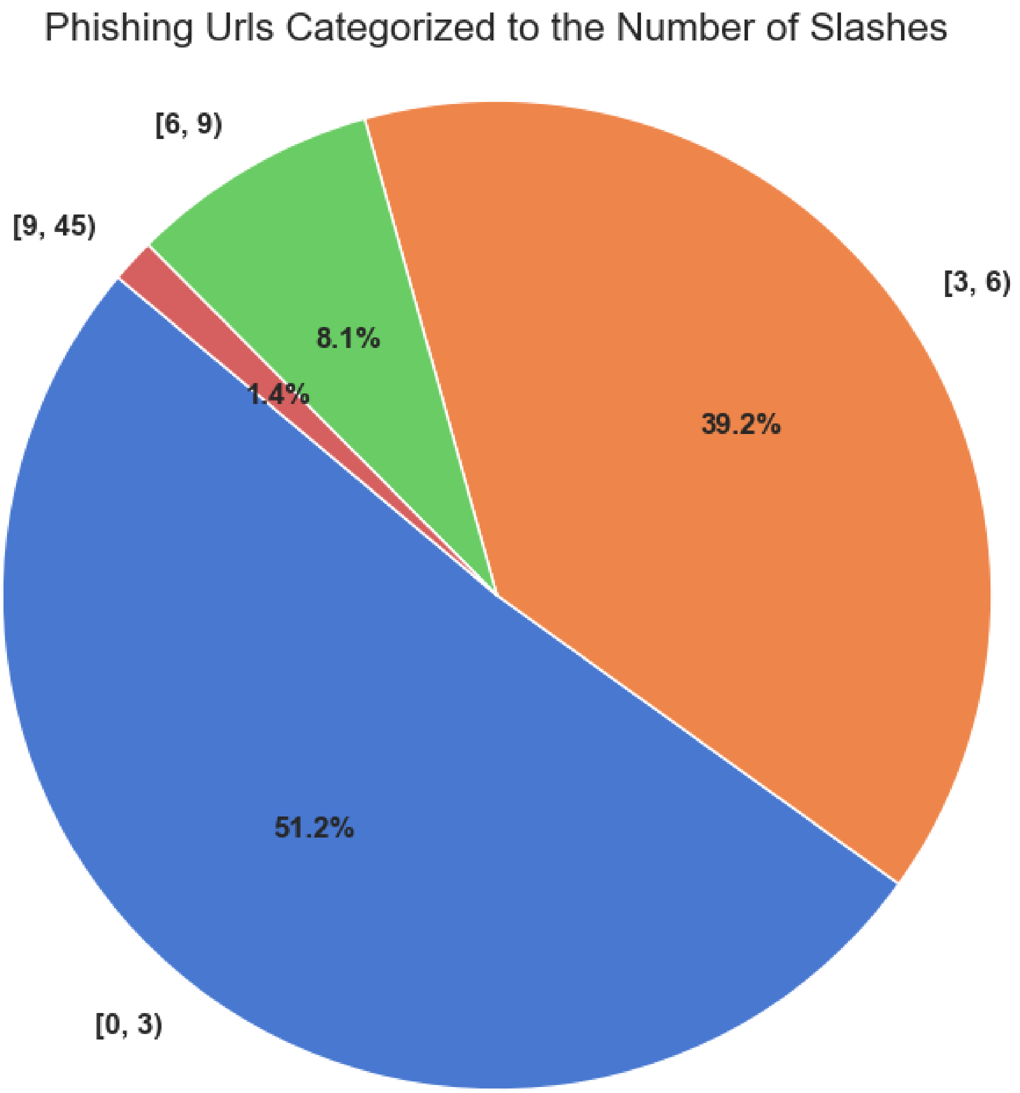


[36]:
```
desired_count = data_model_n_slash_phishing_pivot.iloc[2:, 1].sum()
desired_count
```

[36]:
```
135
```

[39]:
```
pastel_palette = sns.color_palette('muted')

# Data for the pie chart
labels = ['[0, 3)', '[3, 6)', '[6, 9)', '[9, 45)']
sizes = data_model_n_slash_phishing_pivot['phishing_count'].loc[0:3] +␣
  ↪[desired_count]

# Use pastel colors for the pie chart
colors = pastel_palette[:len(labels)]  # Use as many colors as there are labels

# Create the pie chart
```

```
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140, textprops={'fontsize': 12, 'fontweight': 'bold'})

# Set title
plt.title('Phishing Urls Categorized to the Number of Slashes', fontsize=16)

# Equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')


plt.savefig("phishing_urls_categorized_to_the_number_of_lashes.jpeg")
```



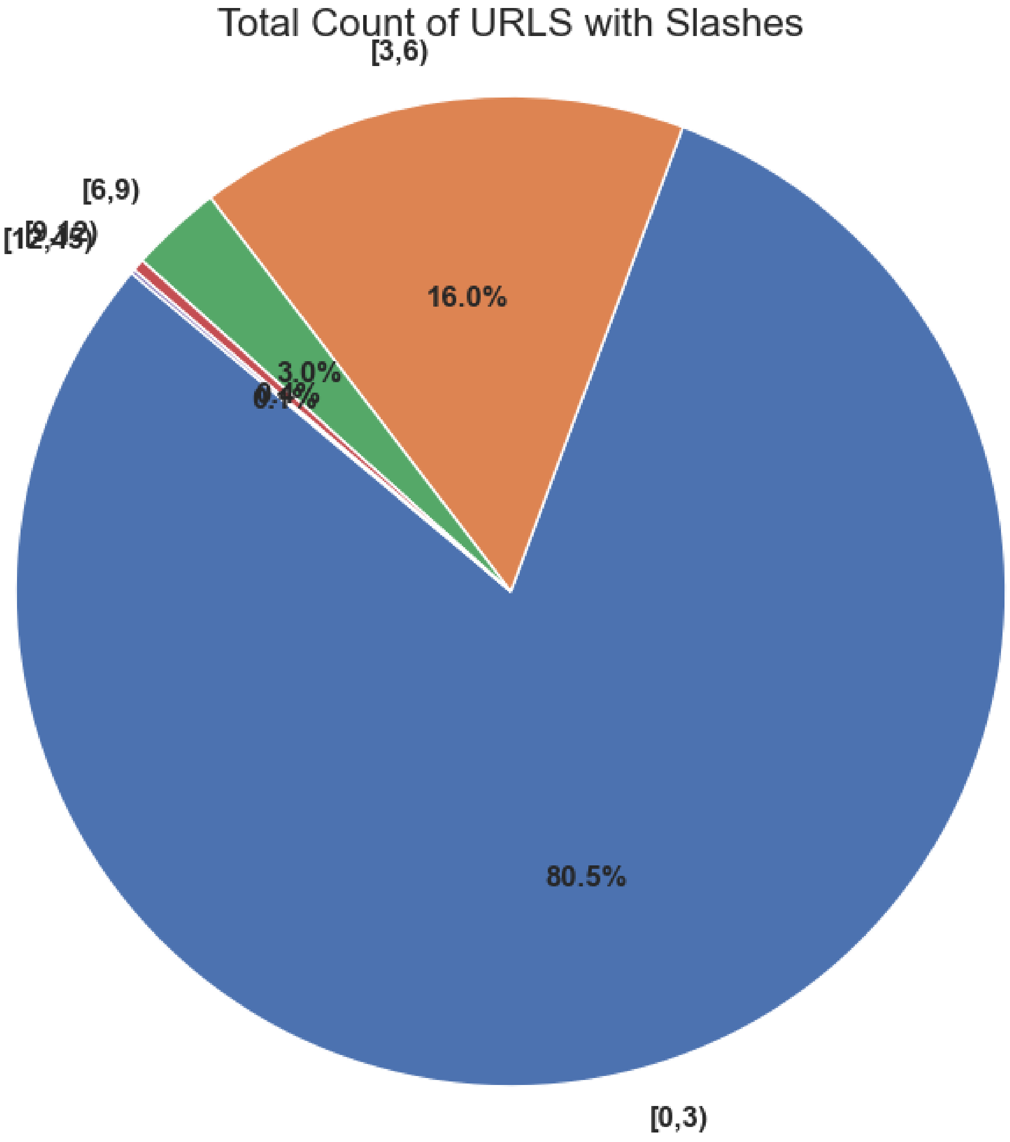
[40]:
```
total_count_df =␣
 ↪data_model_n_slash_phishing_pivot[data_model_n_slash_phishing_pivot['total_count']!
 ↪= 0]
total_count_df
```

[40]:
```
    n_slash_bin  non_phishing_count  phishing_count  total_count  \
0        [0, 3)               61839           18694        80533
1        [3, 6)                1741           14292        16033
2        [6, 9)                 130            2862         2992
3       [9, 12)                   5             390          395
4      [12, 15)                   0              94           94
5      [15, 18)                   0              22           22
6      [18, 21)                   0               4            4
7      [21, 24)                   0               2            2
9      [27, 30)                   0               1            1
14     [42, 45)                   0               1            1

    phishing_percent  non_phishing_percent  rounded_phishing_percent
0          23.212844             76.787156                        23
1          89.141146             10.858854                        89
2          95.655080              4.344920                        96
3          98.734177              1.265823                        99
4         100.000000              0.000000                       100
5         100.000000              0.000000                       100
6         100.000000              0.000000                       100
7         100.000000              0.000000                       100
9         100.000000              0.000000                       100
14        100.000000              0.000000                       100
```

[41]:
```
desired_count = total_count_df.iloc[2:, 1].sum()
```

[43]:
```
pastel_palette = sns.color_palette('deep')

# Data for the pie chart
labels = ['[0,3)', '[3,6)' , '[6,9)', '[9,12)' , '[12,45)']
sizes = list(total_count_df['total_count'][:4]) + [desired_count]

labels

# Use pastel colors for the pie chart
colors = pastel_palette[:len(labels)]  # Use as many colors as there are labels

# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140,␣
 ↪textprops={'fontsize': 12, 'fontweight': 'bold'})
```

```
# Set title
plt.title('Total Count of URLS with Slashes', fontsize=16)

# Equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')

plt.savefig("total_count_urls_to_the_number_of_lashes.jpeg")
```



```
[50]: data_model_n_redirection_df = data.groupby(['n_redirection', 'phishing'],␣
       ↪observed=False).size()

      # Reset the index to flatten the DataFrame
```

```
data_model_n_redirection_df = data_model_n_redirection_df.
 ↪reset_index(name='count')

bins = list(range(-1, data_model_n_redirection_df['n_redirection'].max() + 5,␣
 ↪3))  # Create bins of size 4, starting from 0
data_model_n_redirection_df['n_redirection_bin'] = pd.
 ↪cut(data_model_n_redirection_df['n_redirection'], bins=bins, right=False)
redirection_phishing_df =␣
 ↪data_model_n_redirection_df[data_model_n_redirection_df['phishing'] == 1].
 ↪groupby('n_redirection_bin').sum().reset_index()
non_ection_phishing_df =␣
 ↪data_model_n_redirection_df[data_model_n_redirection_df['phishing'] == 0].
 ↪groupby('n_redirection_bin').sum().reset_index()
```

```
C:\Users\BRINDHA\AppData\Local\Temp\ipykernel_19424\356151943.py:8:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  redirection_phishing_df =
data_model_n_redirection_df[data_model_n_redirection_df['phishing'] ==
1].groupby('n_redirection_bin').sum().reset_index()
C:\Users\BRINDHA\AppData\Local\Temp\ipykernel_19424\356151943.py:9:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  non_ection_phishing_df =
data_model_n_redirection_df[data_model_n_redirection_df['phishing'] ==
0].groupby('n_redirection_bin').sum().reset_index()
```

[51]:

```
data_model_n_redirection_df
```

[51]:

```
    n_redirection  phishing  count n_redirection_bin
0              -1         0   5395           [-1, 2)
1              -1         1   1554           [-1, 2)
2               0         0  33856           [-1, 2)
3               0         1  24849           [-1, 2)
4               1         0  19798           [-1, 2)
5               1         1   7858           [-1, 2)
6               2         0   3803            [2, 5)
7               2         1   1572            [2, 5)
8               3         0    628            [2, 5)
9               3         1    382            [2, 5)
10              4         0    180            [2, 5)
11              4         1    115            [2, 5)
12              5         0     38            [5, 8)
13              5         1     19            [5, 8)
14              6         0      7            [5, 8)
```

```
15          6     1     8          [5, 8)
16          7     0     3          [5, 8)
17          7     1     1          [5, 8)
18          8     0     3         [8, 11)
19          9     0     1         [8, 11)
20          9     1     2         [8, 11)
21         10     0     1         [8, 11)
22         10     1     1         [8, 11)
23         11     0     1        [11, 14)
24         12     0     1        [11, 14)
25         17     1     1        [17, 20)
```

```
[52]: redirection_phishing_df
```

```
[52]:   n_redirection_bin  n_redirection  phishing  count
      0           [-1, 2)              0         3  34261
      1            [2, 5)              9         3   2069
      2            [5, 8)             18         3     28
      3           [8, 11)             19         2      3
      4          [11, 14)              0         0      0
      5          [14, 17)              0         0      0
      6          [17, 20)             17         1      1
```

```
[53]: non_ection_phishing_df
```

```
[53]:   n_redirection_bin  n_redirection  phishing  count
      0           [-1, 2)              0         0  59049
      1            [2, 5)              9         0   4611
      2            [5, 8)             18         0     48
      3           [8, 11)             27         0      5
      4          [11, 14)             23         0      2
      5          [14, 17)              0         0      0
      6          [17, 20)              0         0      0
```

```
[55]: columns = {
          'n_redirection_bin' : non_ection_phishing_df['n_redirection_bin'],
          'phishing_count': redirection_phishing_df['count'],
          'non_phishing_count' : non_ection_phishing_df['count'],
          'total_count' : redirection_phishing_df['count'] +␣
        ↪non_ection_phishing_df['count'],
          'phishing_percent' : (redirection_phishing_df['count']*100 /␣
        ↪(redirection_phishing_df['count'] + non_ection_phishing_df['count'])).
        ↪fillna(0),
          'non_phishing_percent' : 100*(1- (redirection_phishing_df['count'] /␣
        ↪(redirection_phishing_df['count'] + non_ection_phishing_df['count']))).
        ↪fillna(0)
      }
```

```
n_redirection_table_df = pd.DataFrame(columns)
```

[77]:
```
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Model

# Define input layer
input_layer = Input(shape=(19,))

# Define hidden layers
hidden1 = Dense(64, activation='relu')(input_layer)
hidden2 = Dense(32, activation='relu')(hidden1)

# Define output layer
output_layer = Dense(1, activation='sigmoid')(hidden2)

# Create the model
model = Model(inputs=input_layer, outputs=output_layer)

# Display model summary
model.summary()
```

```
Model: "functional_10"


 Layer (type)                          Output Shape                          ␣
↪Param #

 input_layer_3 (InputLayer)            (None, 19)                                ␣
↪  0

 dense_9 (Dense)                       (None, 64)                              ␣
↪1,280

 dense_10 (Dense)                      (None, 32)                              ␣
↪2,080

 dense_11 (Dense)                      (None, 1)                                 ␣
↪ 33



 Total params: 3,393 (13.25 KB)


 Trainable params: 3,393 (13.25 KB)


 Non-trainable params: 0 (0.00 B)
```

[56]:

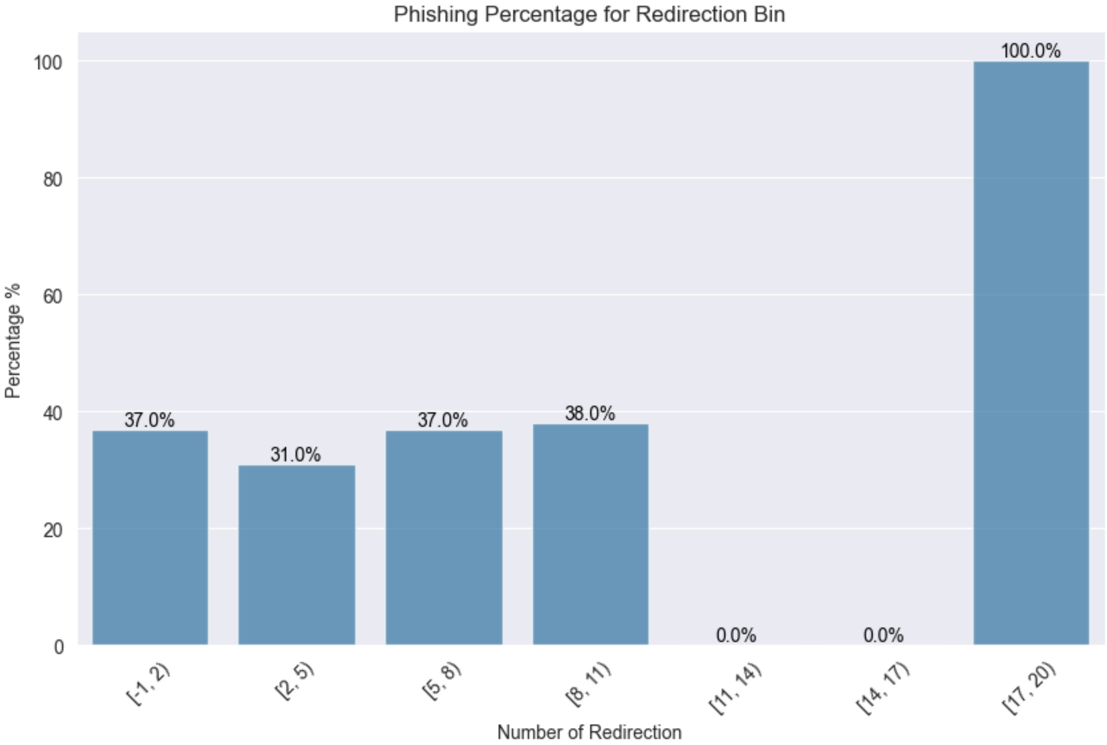
```
n_redirection_table_df
```

[56]:

```
  n_redirection_bin  phishing_count  non_phishing_count  total_count  \
0           [-1, 2)           34261               59049        93310
1            [2, 5)            2069                4611         6680
2            [5, 8)              28                  48           76
3           [8, 11)               3                   5            8
4          [11, 14)               0                   2            2
5          [14, 17)               0                   0            0
6          [17, 20)               1                   0            1

   phishing_percent  non_phishing_percent
0         36.717394             63.282606
1         30.973054             69.026946
2         36.842105             63.157895
3         37.500000             62.500000
4          0.000000            100.000000
5          0.000000              0.000000
6        100.000000              0.000000
```

[57]:

```
sns.set_style("darkgrid")

# Set the width of the bars
bar_width = 0.8  # Adjust this value to increase or decrease the width of the␣
 ↪bars

# Round phishing percentages to the nearest integer
n_redirection_table_df['rounded_phishing_percent'] =␣
 ↪n_redirection_table_df['phishing_percent'].round().astype(int)

# Create the bar chart
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='n_redirection_bin', y='rounded_phishing_percent',␣
 ↪data=n_redirection_table_df, errorbar=None, alpha=0.7, width=bar_width)

# Annotate percentages on top of each bar
for p in ax.patches:
    ax.annotate(f'{p.get_height()}%', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='center', fontsize=10, color='black',␣
 ↪xytext=(0, 5),
                textcoords='offset points')

# Set labels and title
plt.xlabel('Number of Redirection')
plt.ylabel('Percentage %')
plt.title('Phishing Percentage for Redirection Bin')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.savefig("phishing_percentage_for_redirection.jpeg")
```



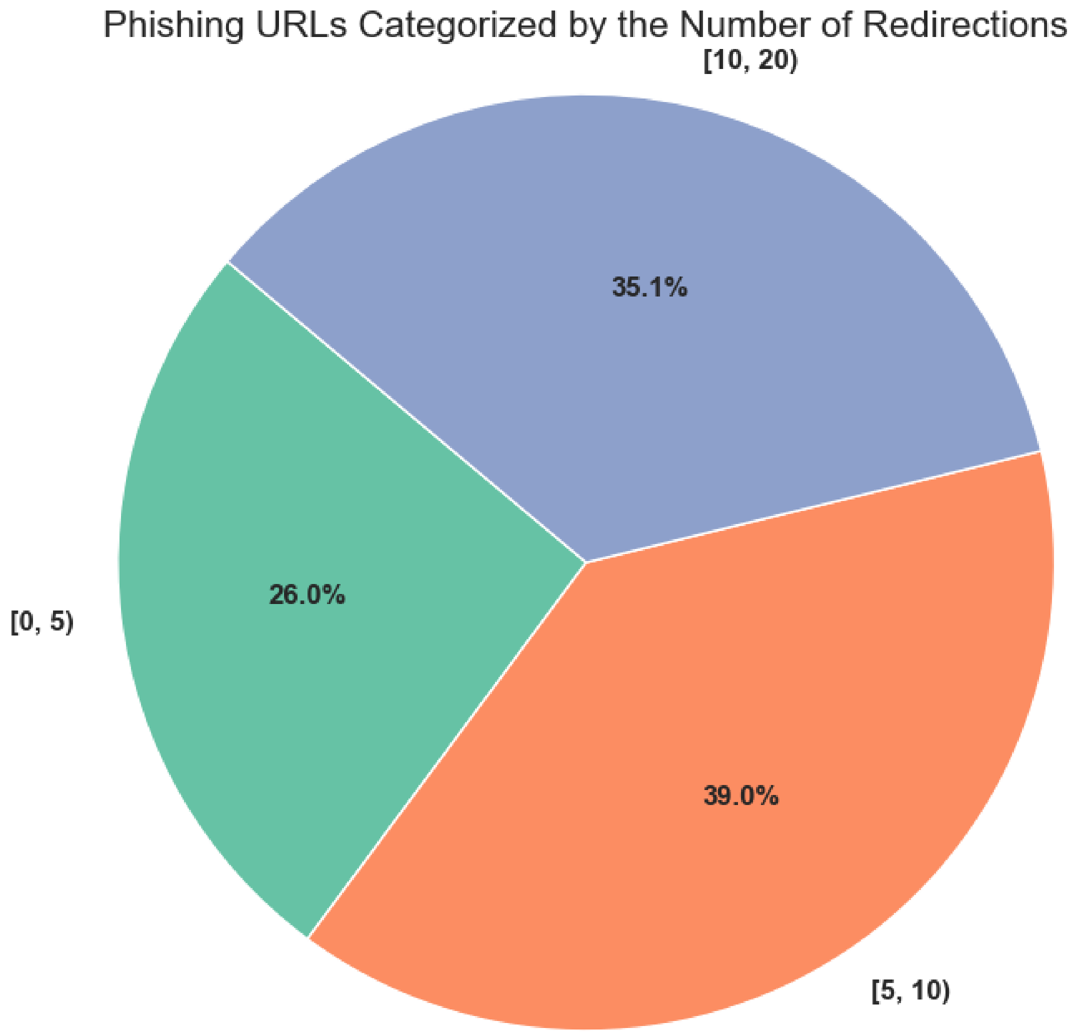
```
[58]: pie_bar_df =  n_redirection_table_df[n_redirection_table_df['phishing_count'] !
        ↪= 0].reset_index(drop=True)
      pie_bar_df
```

```
[58]:   n_redirection_bin  phishing_count  non_phishing_count  total_count  \
      0           [-1, 2)           34261               59049        93310
      1            [2, 5)            2069                4611         6680
      2            [5, 8)              28                  48           76
      3           [8, 11)               3                   5            8
      4          [17, 20)               1                   0            1

         phishing_percent  non_phishing_percent  rounded_phishing_percent
      0         36.717394             63.282606                        37
      1         30.973054             69.026946                        31
```

```
2        36.842105           63.157895                       37
3        37.500000           62.500000                       38
4       100.000000            0.000000                      100
```

[59]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Define a color palette (Set2)
color_palette = sns.color_palette('Set2')

# Data for the pie chart
labels = ['[0, 5)', '[5, 10)', '[10, 20)']  # Update labels accordingly
sizes = [100, 150, desired_count]  # Update sizes accordingly

# Use custom colors for the pie chart
colors = color_palette[:len(labels)]  # Use as many colors as there are labels

# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140,␣
 ↪textprops={'fontsize': 12, 'fontweight': 'bold'})

# Set title
plt.title('Phishing URLs Categorized by the Number of Redirections',␣
 ↪fontsize=16)

# Equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')

# Save the plot as a JPEG file
plt.savefig("phishing_urls_categorized_to_the_number_of_redirection.jpeg")

plt.show()
```



[61]:
```
import pandas as pd

# Group by 'url_length' and 'phishing', and count occurrences
data_model_url_length = data.groupby(['url_length', 'phishing'],␣
 ↪observed=False).size()

# Reset index to create a DataFrame
data_url_length_df =  data_model_url_length.reset_index(name='count')

# Define bins for 'url_length'
bins = list(range(-1, data_url_length_df['url_length'].max() + 101, 100))  #␣
 ↪Create bins of size 100, starting from -1
```

```
# Create a new column with bin labels
data_url_length_df['url_length_bin'] = pd.cut(data_url_length_df['url_length'],␣
 ↪bins=bins, right=False)

# Group by 'url_length_bin' and sum counts for phishing URLs
phishing_df = data_url_length_df[data_url_length_df['phishing'] == 1].
 ↪groupby('url_length_bin').sum().reset_index()

# Group by 'url_length_bin' and sum counts for non-phishing URLs
non_phishing_df = data_url_length_df[data_url_length_df['phishing'] == 0].
 ↪groupby('url_length_bin').sum().reset_index()
```

```
C:\Users\BRINDHA\AppData\Local\Temp\ipykernel_19424\236722338.py:16:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  phishing_df = data_url_length_df[data_url_length_df['phishing'] ==
1].groupby('url_length_bin').sum().reset_index()
C:\Users\BRINDHA\AppData\Local\Temp\ipykernel_19424\236722338.py:19:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  non_phishing_df = data_url_length_df[data_url_length_df['phishing'] ==
0].groupby('url_length_bin').sum().reset_index()
```

[62]:

```
phishing_df.url_length_bin == non_phishing_df.url_length_bin
```

[62]:

```
0      True
1      True
2      True
3      True
4      True
5      True
6      True
7      True
8      True
9      True
10     True
11     True
12     True
13     True
14     True
15     True
16     True
17     True
18     True
19     True
```

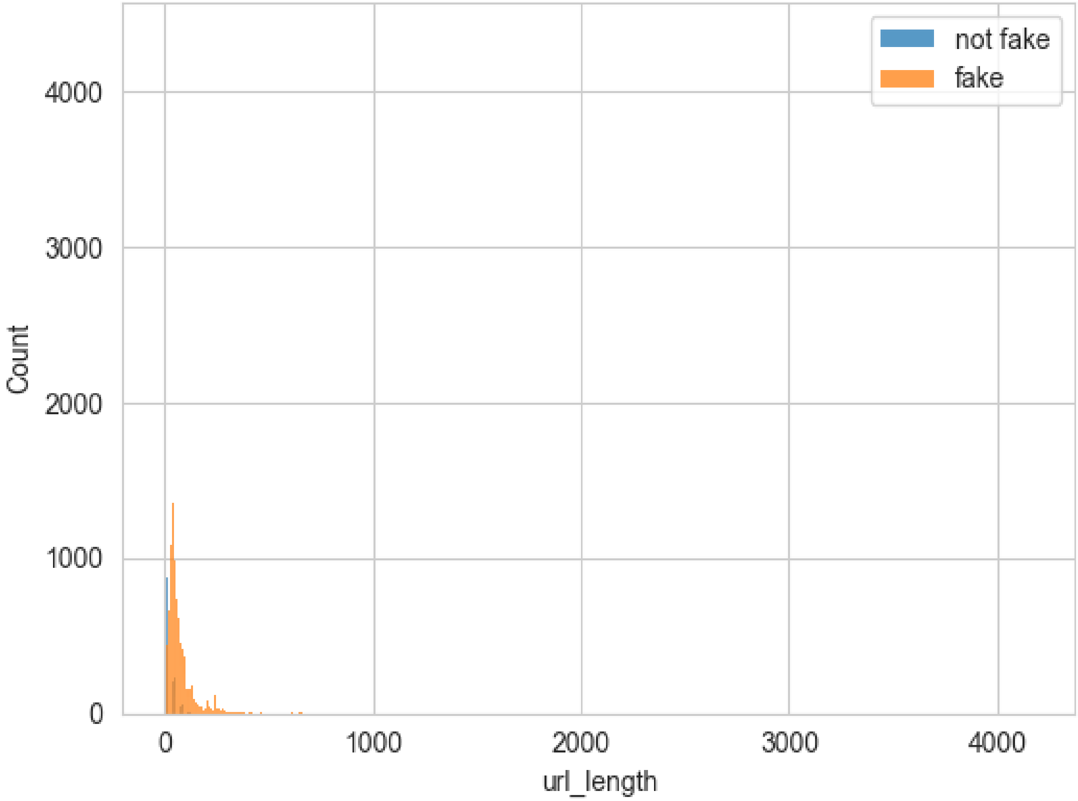
```
20    True
21    True
22    True
23    True
24    True
25    True
26    True
27    True
28    True
29    True
30    True
31    True
32    True
33    True
34    True
35    True
36    True
37    True
38    True
39    True
40    True
41    True
Name: url_length_bin, dtype: bool
```

[63]:

```
import pandas as pd

# Define the columns using dictionaries
columns = {
    'url_length_bin': phishing_df['url_length_bin'],
    'phishing_count': phishing_df['count'],
    'non_phishing_count': non_phishing_df['count'],
    'total_count': phishing_df['count'] + non_phishing_df['count'],
    'phishing_percent': (phishing_df['count'] * 100 / (phishing_df['count'] +␣
 ↪non_phishing_df['count'])).fillna(0),
    'non_phishing_percent': 100 * (1 - (phishing_df['count'] /␣
 ↪(phishing_df['count'] + non_phishing_df['count']))).fillna(0)
}

# Create a DataFrame from the columns
n_url_length_table_df = pd.DataFrame(columns)
```

[72]:

```
sns.histplot(data[data['phishing']==0]['url_length'],label='not fake')
sns.histplot(data[data['phishing']==1]['url_length'],label='fake')
plt.legend()
plt.show()
```



```
[64]: n_url_length_table_df
```

```
[64]:    url_length_bin  phishing_count  non_phishing_count  total_count  \
      0         [-1, 99)           30718               63226        93944
      1        [99, 199)            3754                 474         4228
      2       [199, 299)            1562                   8         1570
      3       [299, 399)             184                   1          185
      4       [399, 499)              67                   0           67
      5       [499, 599)              28                   6           34
      6       [599, 699)              16                   0           16
      7       [699, 799)              12                   0           12
      8       [799, 899)               5                   0            5
      9       [899, 999)               7                   0            7
      10     [999, 1099)               1                   0            1
      11    [1099, 1199)               0                   0            0
      12    [1199, 1299)               2                   0            2
      13    [1299, 1399)               3                   0            3
      14    [1399, 1499)               0                   0            0
      15    [1499, 1599)               0                   0            0
      16    [1599, 1699)               1                   0            1
      17    [1699, 1799)               0                   0            0
```

```
18   [1799, 1899)                0                   0            0
19   [1899, 1999)                1                   0            1
20   [1999, 2099)                0                   0            0
21   [2099, 2199)                0                   0            0
22   [2199, 2299)                0                   0            0
23   [2299, 2399)                0                   0            0
24   [2399, 2499)                0                   0            0
25   [2499, 2599)                0                   0            0
26   [2599, 2699)                0                   0            0
27   [2699, 2799)                0                   0            0
28   [2799, 2899)                0                   0            0
29   [2899, 2999)                0                   0            0
30   [2999, 3099)                0                   0            0
31   [3099, 3199)                0                   0            0
32   [3199, 3299)                0                   0            0
33   [3299, 3399)                0                   0            0
34   [3399, 3499)                0                   0            0
35   [3499, 3599)                0                   0            0
36   [3599, 3699)                0                   0            0
37   [3699, 3799)                0                   0            0
38   [3799, 3899)                0                   0            0
39   [3899, 3999)                0                   0            0
40   [3999, 4099)                0                   0            0
41   [4099, 4199)                1                   0            1

    phishing_percent  non_phishing_percent
0          32.698203             67.301797
1          88.789026             11.210974
2          99.490446              0.509554
3          99.459459              0.540541
4         100.000000              0.000000
5          82.352941             17.647059
6         100.000000              0.000000
7         100.000000              0.000000
8         100.000000              0.000000
9         100.000000              0.000000
10        100.000000              0.000000
11          0.000000              0.000000
12        100.000000              0.000000
13        100.000000              0.000000
14          0.000000              0.000000
15          0.000000              0.000000
16        100.000000              0.000000
17          0.000000              0.000000
18          0.000000              0.000000
19        100.000000              0.000000
20          0.000000              0.000000
```

```
21         0.000000           0.000000
22         0.000000           0.000000
23         0.000000           0.000000
24         0.000000           0.000000
25         0.000000           0.000000
26         0.000000           0.000000
27         0.000000           0.000000
28         0.000000           0.000000
29         0.000000           0.000000
30         0.000000           0.000000
31         0.000000           0.000000
32         0.000000           0.000000
33         0.000000           0.000000
34         0.000000           0.000000
35         0.000000           0.000000
36         0.000000           0.000000
37         0.000000           0.000000
38         0.000000           0.000000
39         0.000000           0.000000
40         0.000000           0.000000
41       100.000000           0.000000
```

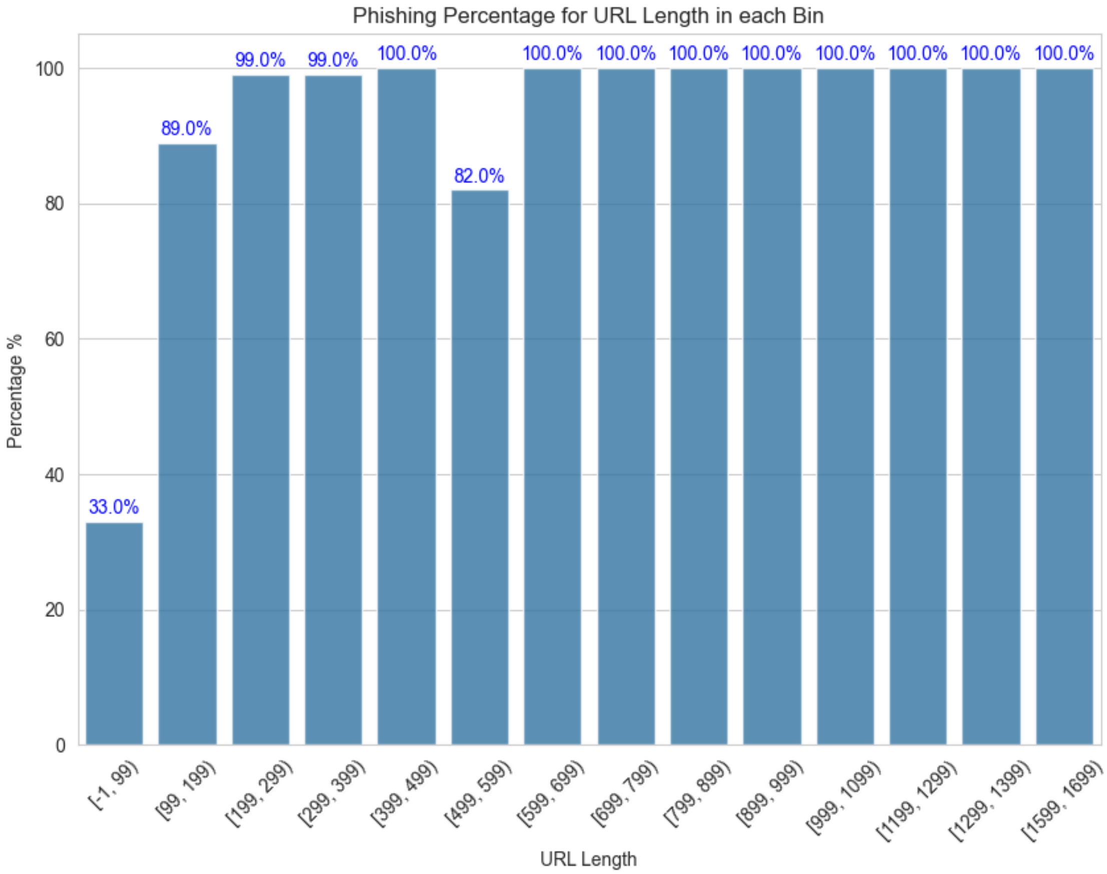
[65]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the style
sns.set_style("whitegrid")

# Set the width of the bars
bar_width = 0.8  # Adjust this value to increase or decrease the width of the␣
 ↪bars

# Filter out rows where either phishing_count or non_phishing_count is zero
n_url_length_table_df =␣
 ↪n_url_length_table_df[(n_url_length_table_df['phishing_count'] != 0) |␣
 ↪(n_url_length_table_df['non_phishing_count'] != 0)].reset_index()

# Round phishing percentages to the nearest integer
n_url_length_table_df['rounded_phishing_percent'] =␣
 ↪n_url_length_table_df['phishing_percent'].round().astype(int)
```

[68]:

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 7))

# Get the data for plotting
```

```python
x = list(n_url_length_table_df['url_length_bin'][:14])
y = n_url_length_table_df['rounded_phishing_percent'][:14]

# Create the bar plot
ax = sns.barplot(x=x, y=y, errorbar=None, alpha=0.8, width=bar_width)

# Annotate percentages on top of each bar
for p in ax.patches:
    ax.annotate(f'{p.get_height()}%', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=10, color='blue', xytext=(0, 7),
                textcoords='offset points')

# Set labels and title
plt.xlabel('URL Length')
plt.ylabel('Percentage %')
plt.title('Phishing Percentage for URL Length in each Bin')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Save the plot as a JPEG file
plt.savefig("phishing_percentage_url_each_bin.jpeg")

plt.show()
```



```
[69]: n_url_length_table_df
```

```
[69]:     index url_length_bin  phishing_count  non_phishing_count  total_count  \
      0       0       [-1, 99)           30718               63226        93944
      1       1      [99, 199)            3754                 474         4228
      2       2     [199, 299)            1562                   8         1570
      3       3     [299, 399)             184                   1          185
      4       4     [399, 499)              67                   0           67
      5       5     [499, 599)              28                   6           34
      6       6     [599, 699)              16                   0           16
      7       7     [699, 799)              12                   0           12
      8       8     [799, 899)               5                   0            5
      9       9     [899, 999)               7                   0            7
      10     10    [999, 1099)               1                   0            1
      11     12   [1199, 1299)               2                   0            2
      12     13   [1299, 1399)               3                   0            3
      13     16   [1599, 1699)               1                   0            1
      14     19   [1899, 1999)               1                   0            1
      15     41   [4099, 4199)               1                   0            1
```

```
    phishing_percent  non_phishing_percent  rounded_phishing_percent
0          32.698203             67.301797                        33
1          88.789026             11.210974                        89
2          99.490446              0.509554                        99
3          99.459459              0.540541                        99
4         100.000000              0.000000                       100
5          82.352941             17.647059                        82
6         100.000000              0.000000                       100
7         100.000000              0.000000                       100
8         100.000000              0.000000                       100
9         100.000000              0.000000                       100
10        100.000000              0.000000                       100
11        100.000000              0.000000                       100
12        100.000000              0.000000                       100
13        100.000000              0.000000                       100
14        100.000000              0.000000                       100
15        100.000000              0.000000                       100
```

```
[ ]:
```