15_analysis_on_three_word_zero_shot

June 25, 2024

1 Analysis on Three Word Single-shot classification problem

We can setup the project, with the needed functions, libraries and import the data in a useful-kind of object.

After that, we can see that each element has been importet without any problems (15000 in, 15000 out), how the data are created and disposed in the dataframe, some kind of initial information about data distributions and numerical analysis and at the end, the used data-model.

[ ]:
```
import polars as pl
import json
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

[2]:
```
data = []

with open("../data/prepared/zero_shot_classification/result_1.txt", "r") as of:
        lines = of.readlines()
        for line in lines:
                oj = json.loads(line)

                #print(oj.keys()) # dict_keys(['sequence', 'labels', 'scores'])
                oj["sequence"] = json.loads(oj["sequence"])
                oj["undecided"] = oj["scores"][oj["labels"].index("undecided")]␣
 ↪# ["undecided", "malicious", "benign"]
                oj["malicious"] = oj["scores"][oj["labels"].index("malicious")]
                oj["benign"] = oj["scores"][oj["labels"].index("benign")]

                oj["max_value"] = max(oj["scores"])
                oj["output"] = oj["labels"][oj["scores"].index(oj["max_value"])]

                data.append(oj)

len(data)
```

[2]:
```
15000
```

[27]:
```
df = pl.DataFrame(data)
df.head(2)
```

[27]: shape: (2, 8)

```
  sequence     labels       scores       undecided   malicious   benign
max_value   output
  ---          ---          ---          ---         ---         ---         ---
  ---
  struct[1]    list[str]    list[f64]    f64         f64         f64         f64
  str


  {"{"userAge  ["undecide   [0.795721,   0.795721    0.127998    0.076281
0.795721    undecided
  nt": "Boto3  d", "malic   0.127998,

  /1.9.201 P…  ious",       0.076281]

               "beni…

  {"{"userAge  ["undecide   [0.794392,   0.794392    0.098194    0.107414
0.794392    undecided
  nt": "Boto3  d",          0.107414,

  /1.9.201 P…  "benign",    0.098194]

               "malicio…
```

[30]:
```
df.describe()
```

[30]: shape: (9, 9)

```
  statistic   sequence   labels    scores    …   malicious   benign
max_value   output
  ---         ---        ---       ---           ---         ---         ---
  ---
  str         f64        f64       f64           f64         f64         f64
  str


  count       15000.0    15000.0   15000.0   …   15000.0     15000.0     15000.0
```

```
 15000
 null_count   0.0        0.0        0.0        …   0.0         0.0        0.0
 0
 mean          null       null       null       …   0.14499     0.144968
0.711823   null
 std           null       null       null       …   0.080772    0.105572   0.17874
 null
 min           null       null       null       …   0.031416    0.025201
0.333451   benign
 25%           null       null       null       …   0.097757    0.073231
0.736932   null
 50%           null       null       null       …   0.112673    0.104847
0.782085   null
 75%           null       null       null       …   0.145361    0.132741
0.822661   null
 max           null       null       null       …   0.39466     0.609151
0.943383   undecided
```

[29]:
```
df.schema
```

[29]:
```
OrderedDict([('sequence', Struct({'column_0': String})),
             ('labels', List(String)),
             ('scores', List(Float64)),
             ('undecided', Float64),
             ('malicious', Float64),
             ('benign', Float64),
             ('max_value', Float64),
             ('output', String)])
```

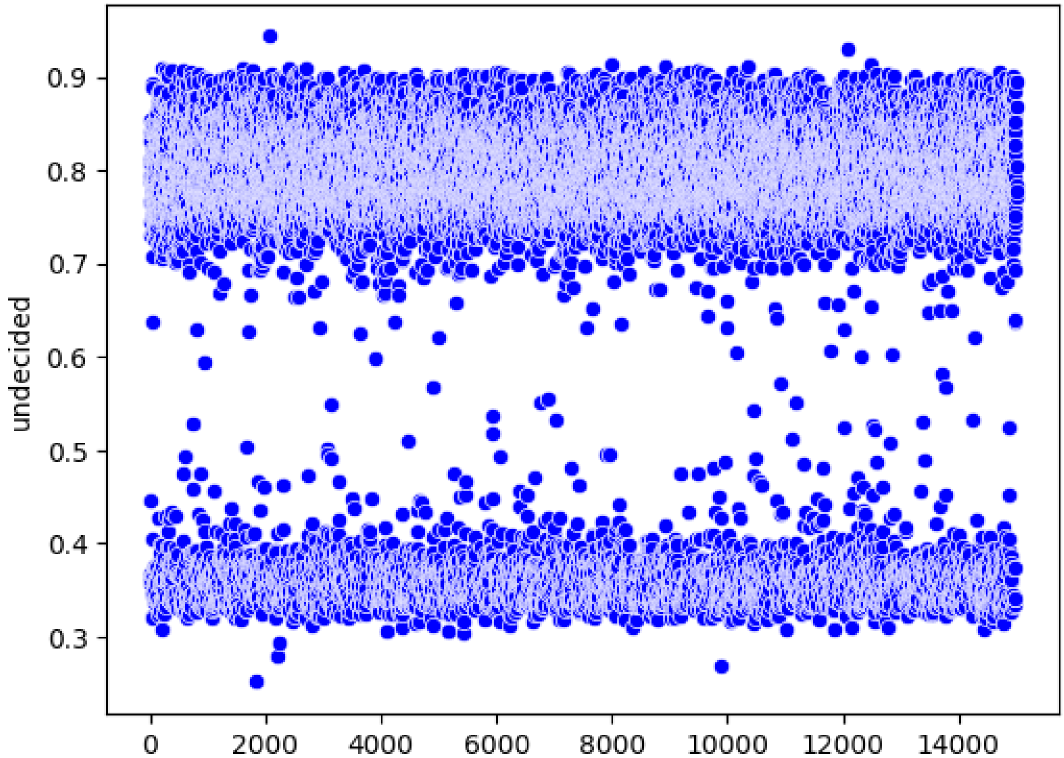
1.1 Study of the Models and Distributions

We can now study:

- first the distribution of the single things taken alone (“undefined”, “malicious” and “benign”).
- after that, we elaborate on the scatterplot/pairplot

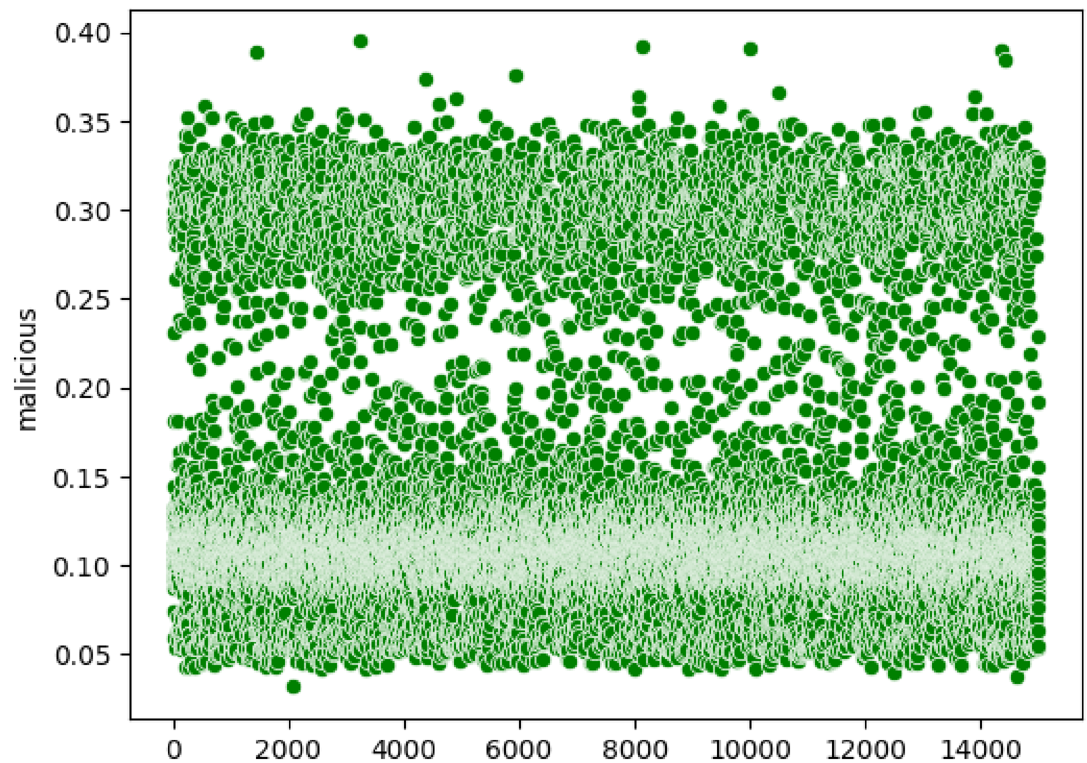
[35]:
```
sns.scatterplot(data=df.to_pandas()["undecided"], color="blue")
```

[35]:
```
<Axes: ylabel='undecided'>
```



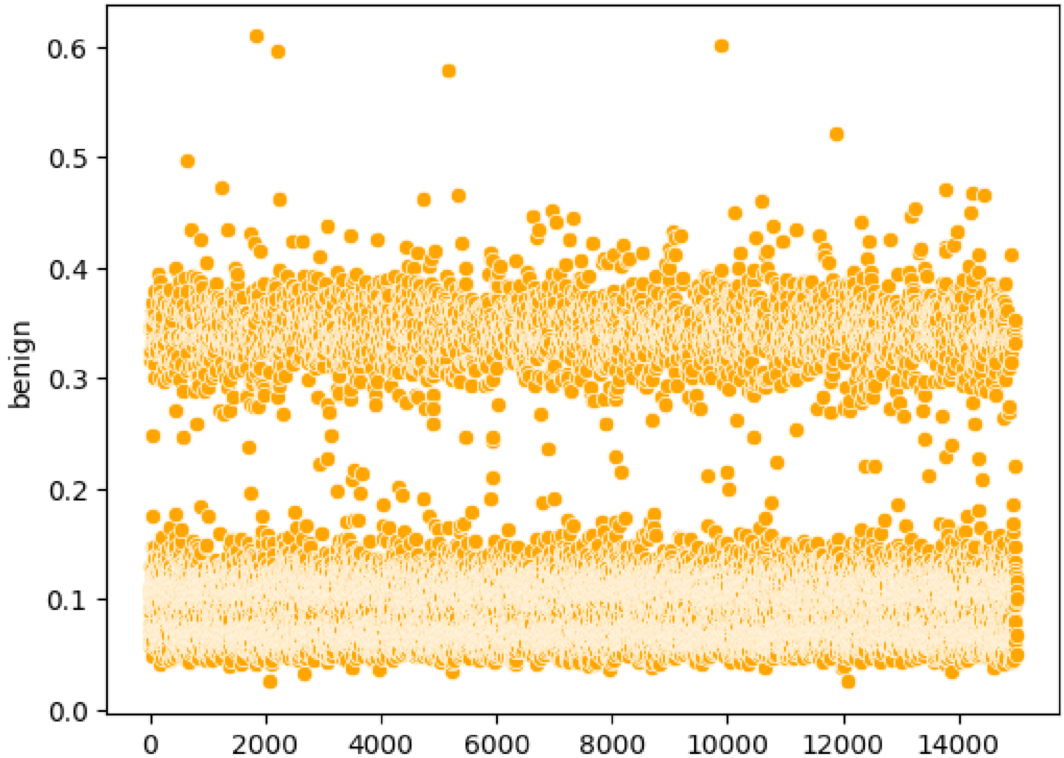
[34]:
```
sns.scatterplot(data=df.to_pandas()["malicious"], color="green")
```

[34]: <Axes: ylabel='malicious'>



[31]:
```
sns.scatterplot(data=df.to_pandas()["benign"], color="orange")
```

[31]: <Axes: ylabel='benign'>



We can now see and read how much value are the output represented. Before, we can see the table of the three possibilities, and after that the plot of the distribution of samples.

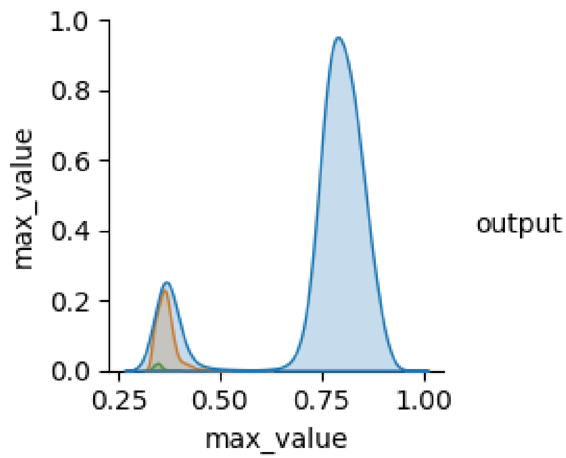
[14]:
```
df["output"].value_counts()
```

[14]: shape: (3, 2)

output	count
---	---
str	u32
benign	1119
malicious	54
undecided	13827

[15]:
```
sns.pairplot(data=df.select("output", "max_value").to_pandas(), hue="output")
```

[15]: <seaborn.axisgrid.PairGrid at 0x7efce8bd97e0>



With this plot, we can see some interesting things:

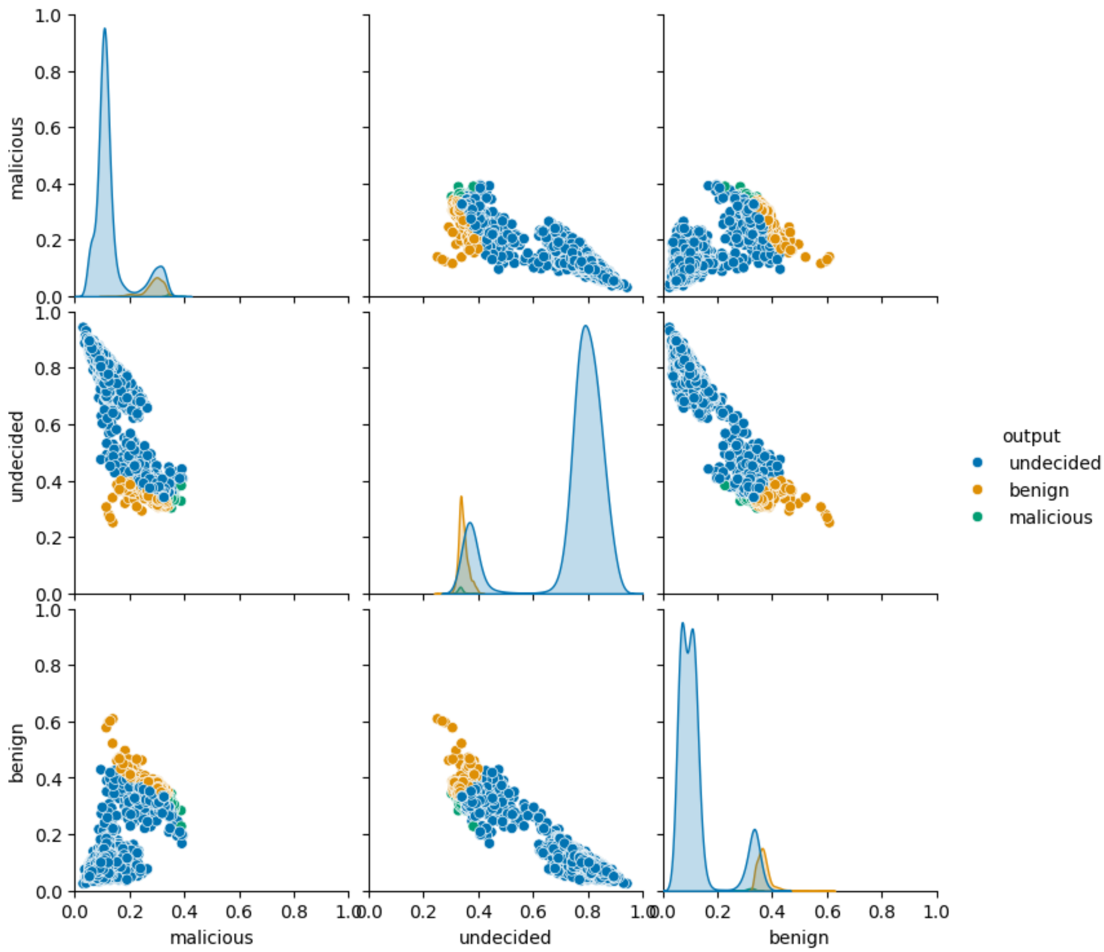
- the "undecided" output has a median value on around 80%, instead of the "malicious" and the "benign" are around 30%.
- the point of interest is the quadrant (0.40, 0.40), in which are concentrated the output equal to "malicious" and "benign".
  - this clearly tells us a *important story*: when some kind of decision is taken, is never certain, since it always resides around 33%. The three components have not a absolute majority able to make some sort of decision.
  - we can see this better in the second distribution plot: we have a majority of "undecided" around 80%, but around 33$ we have likely the same number of peaks and samples.

Answer: the model **cannot** make a decision.

After that, there are the three single distribution for each of the three possible outputs, the hist-plot of the occurences in the output section.

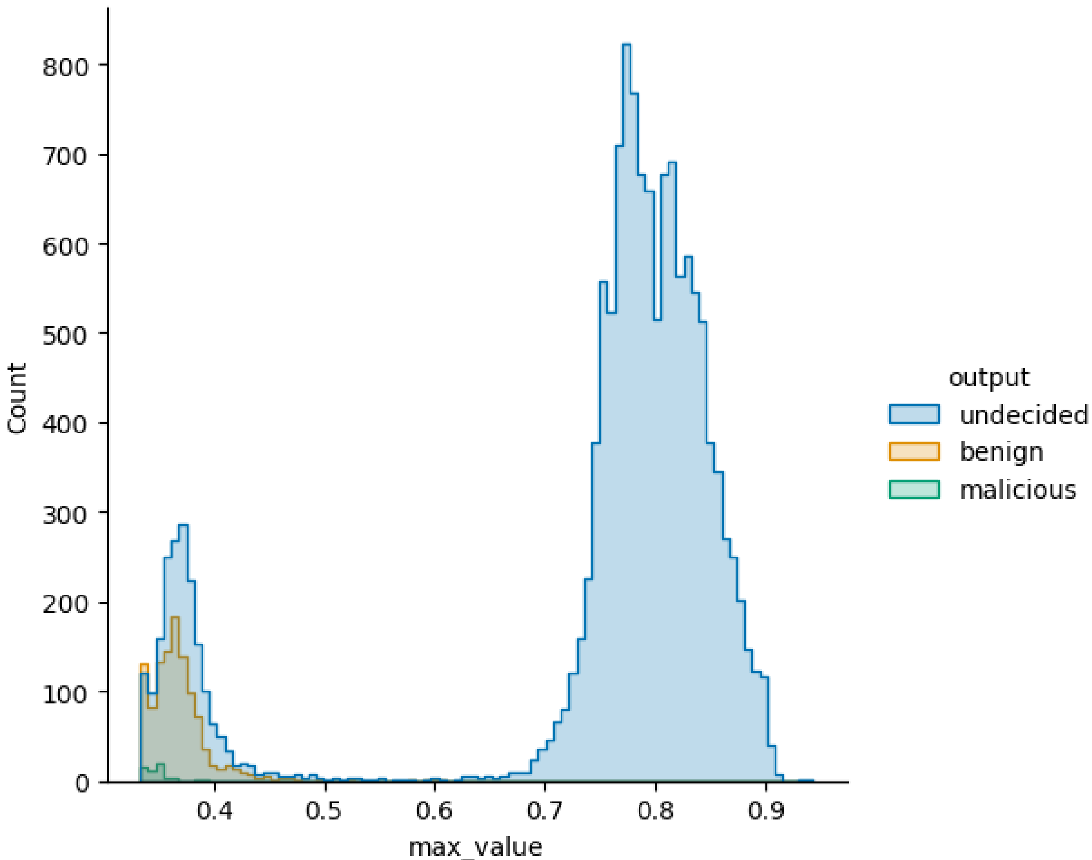
[25]:
```
sns_big_pairplot = sns.pairplot(
        data=df.select("malicious", "undecided", "benign", "output").
 ↪to_pandas(),
        hue="output",
        diag_kind="kde",
        palette=sns.color_palette("colorblind", n_colors=3))
sns_big_pairplot.set(xlim=(0,1), ylim=(0, 1))
print(sns_big_pairplot)
```

```
<seaborn.axisgrid.PairGrid object at 0x7fa05140cb20>
```



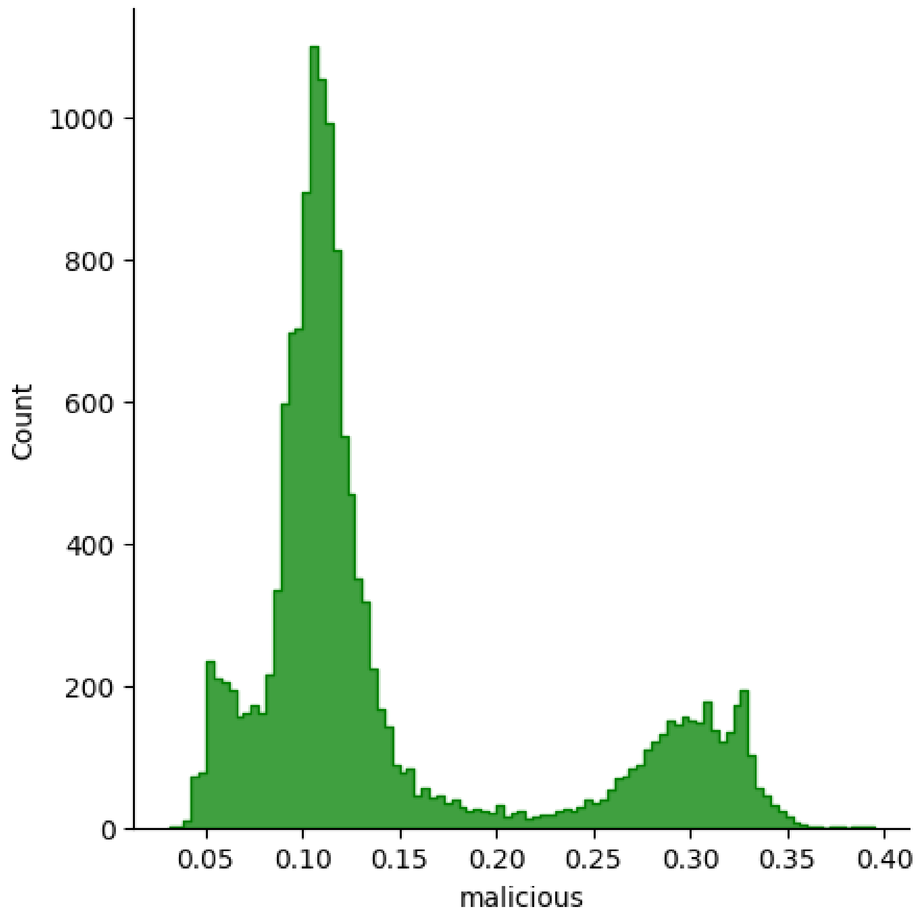
[39]:
```
sns.displot(data=df.to_pandas(), x="max_value", hue="output", element="step",
    palette=sns.color_palette("colorblind", n_colors=3))
```

[39]: `<seaborn.axisgrid.FacetGrid at 0x7f9fe4bc70a0>`



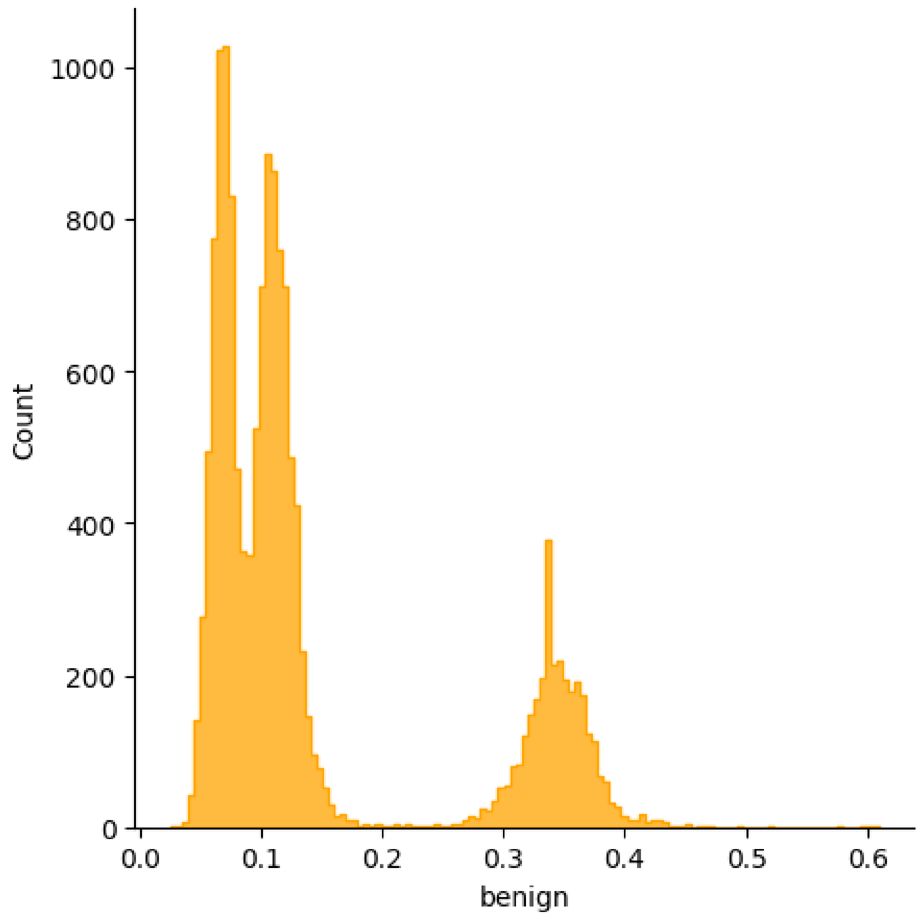
[23]:
```
sns.displot(data=df.to_pandas(), x="malicious", element="step", color="green")
```

[23]: <seaborn.axisgrid.FacetGrid at 0x7efc95a25960>



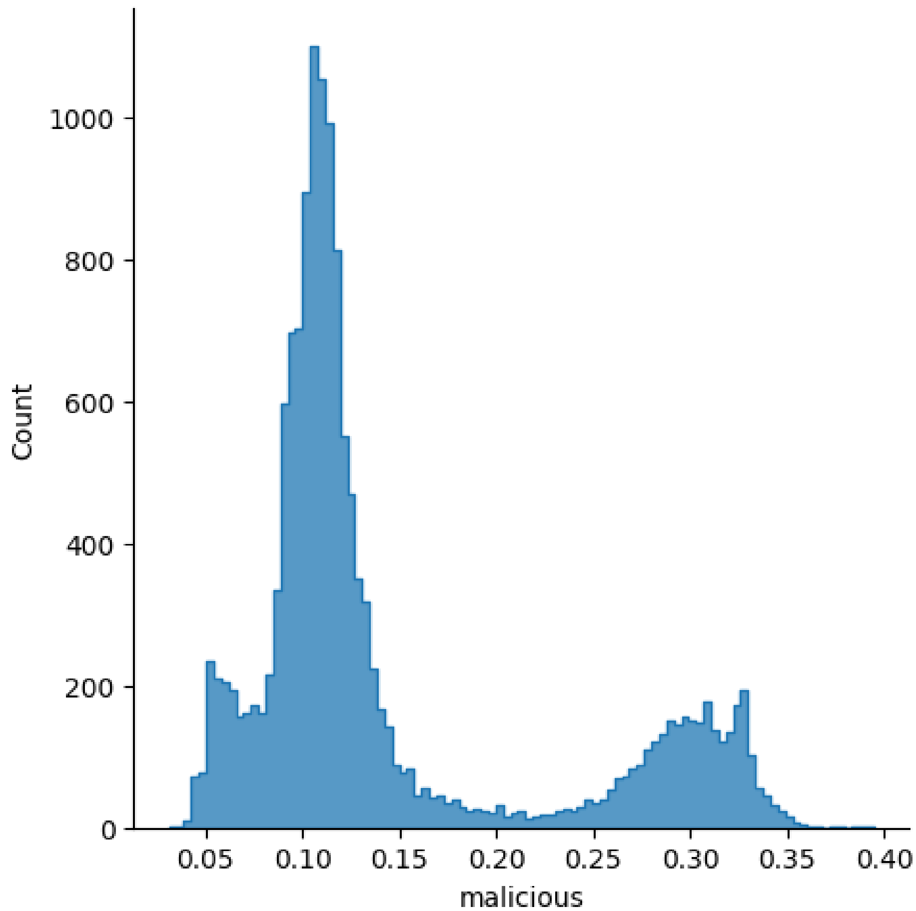
[24]:
```
sns.displot(data=df.to_pandas(), x="benign", element="step", color="orange")
```

[24]: <seaborn.axisgrid.FacetGrid at 0x7efca00c1810>



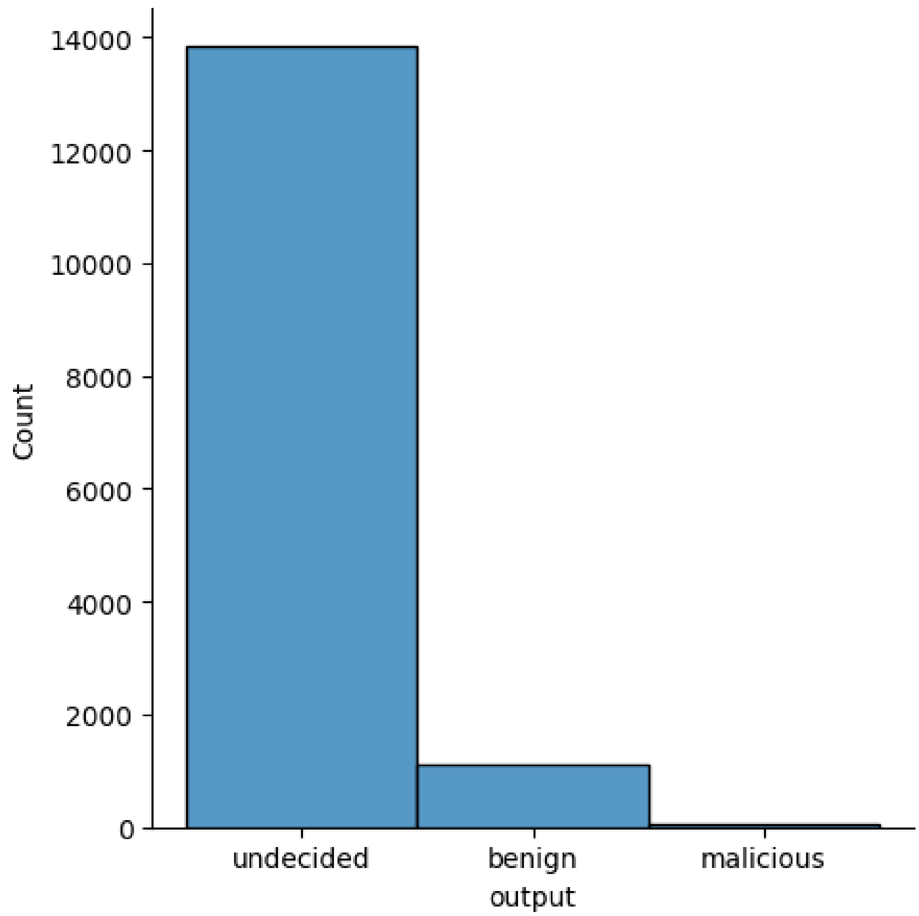
[25]:
```
sns.displot(data=df.to_pandas(), x="malicious", element="step")
```

[25]: <seaborn.axisgrid.FacetGrid at 0x7efceadd2110>



[26]:
```
sns.displot(data=df["output"].to_pandas(), discrete=True)
```

[26]: <seaborn.axisgrid.FacetGrid at 0x7efca0164be0>



1.2 Analysis on what "benign" and "malicious" logs contains

This has only been made for "malicious" and "benign", and cover only "errorMessage" fuse. There are some interesting thing here, too. The logs classified as benign or malicious report the same kind of logs ("you are not authorized to perform this operation"). At the end, being malicious or benign, is the same for the model.

[31]:
```
mal_find.select(pl.col("errorMessage").unique())
```

[31]:
```
shape: (1, 1)

  errorMessage
  ---
  str

  You are not authorized to perfor…
```

[32]:
```python
ben_find = df.filter(
                pl.col("output") == "benign"
        ).select("sequence").unnest("sequence").rename({
            "column_0": "decoded"
        }).to_numpy()

with open("../data/prepared/zero_shot_classification/ben_find.ndjson", "w") as
 ↪mf:
        for i in ben_find:
                json.dump(json.loads(i[0]), mf)
                mf.write("\n")

ben_find = pl.read_ndjson("../data/prepared/zero_shot_classification/ben_find.
 ↪ndjson")
ben_find.head()
```

[32]: shape: (5, 18)

```
  userAgent   eventID     errorMess   userIdent   …   eventVers   eventTime
sharedEve   resource
  ---         ---         age         ity             ion         ---
ntID        s
  str         str         ---         ---             ---         str
---         ---
                          str         struct[8]       str
str         list[str

  uct[3]]


  Boto3/1.9   b955fda0-   You are     {"IAMUser   …   1.05        2019-08-2
null        null
  .201 Pyth   ba9f-4cbf   not autho   ","AIDA9B                   2T07:56:4

  on/2.7.12   -b720-c00   rized to    036HFBHKG                   4Z

  Linu…       a802d…      perfor…     JA09C…

  Boto3/1.9   a06035b-2   You are     {"IAMUser   …   1.05        2019-08-2
null        null
  .201 Pyth   7c2-45b0-   not autho   ","AIDAD0                   3T04:54:2

  on/2.7.12   a535-3bfc   rized to    2GQD0K8TE                   5Z

  Linu…       ef6ea…      perfor…     F7KW1…
```

```
 Boto3/1.9   e27993-5a   You are    {"IAMUser   …   1.05        2019-08-2
null        null
 .201 Pyth   7d-49d7-9   not autho  ","AIDADO                   3T01:49:5

 on/2.7.12   ad2-9a3df   rized to   2GQDOK8TE                   7Z

 Linu…       3797b…      perfor…    F7KW1…

 Boto3/1.9   b865653e1   You are    {"IAMUser   …   1.05        2019-08-2
null        null
 .201 Pyth   -8733-4f2   not autho  ","AIDADO                   3T12:04:1

 on/2.7.12   a-8999-5f   rized to   2GQDOK8TE                   6Z

 Linu…       f7ce5…      perfor…    F7KW1…

 Boto3/1.9   dc3324d4-   You are    {"IAMUser   …   1.05        2019-08-2
null        null
 .201 Pyth   9375-4954   not autho  ","AIDADO                   3T01:45:4

 on/2.7.12   -aea9-799   rized to   2GQDOK8TE                   1Z

 Linu…       dce40…      perfor…    F7KW1…
```

[33]:
```python
ben_find.select(pl.col("errorMessage").unique())
```

[33]:
```
shape: (940, 1)

 errorMessage
 ---
 str

 You are not authorized to perfor…
 You are not authorized to perfor…
 You are not authorized to perfor…
 You are not authorized to perfor…
 You are not authorized to perfor…
 …
 You are not authorized to perfor…
 You are not authorized to perfor…
 You are not authorized to perfor…
 You are not authorized to perfor…
 You are not authorized to perfor…
```

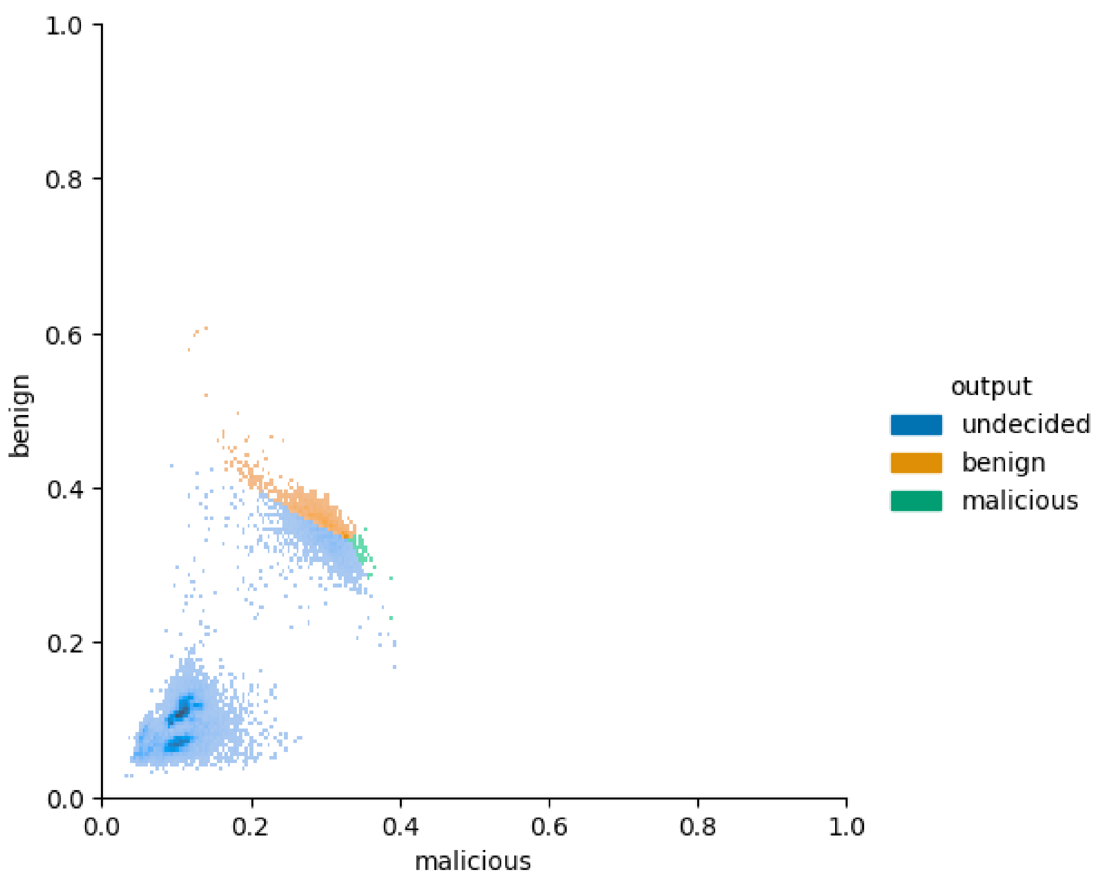
1.3 Check for distribution of the three output together

In this section we recove come of the earlier stage of the analysis, to perform some sort of analysis on how the model disposes the outputs.

In this kind of plots, we can better see that there is not a clear win when the model says "benign" or "malicious", and so there is nothing that clearly reach the decision point.

[44]:
```
sns_plot = sns.displot(
    data=df.select("malicious", "benign", "output").to_pandas(),
    x="malicious",
    y="benign",
        hue="output",
        palette=sns.color_palette("colorblind", n_colors=3)
)
sns_plot.set(ylim=(0,1), xlim=(0,1))
print(sns_plot)
```

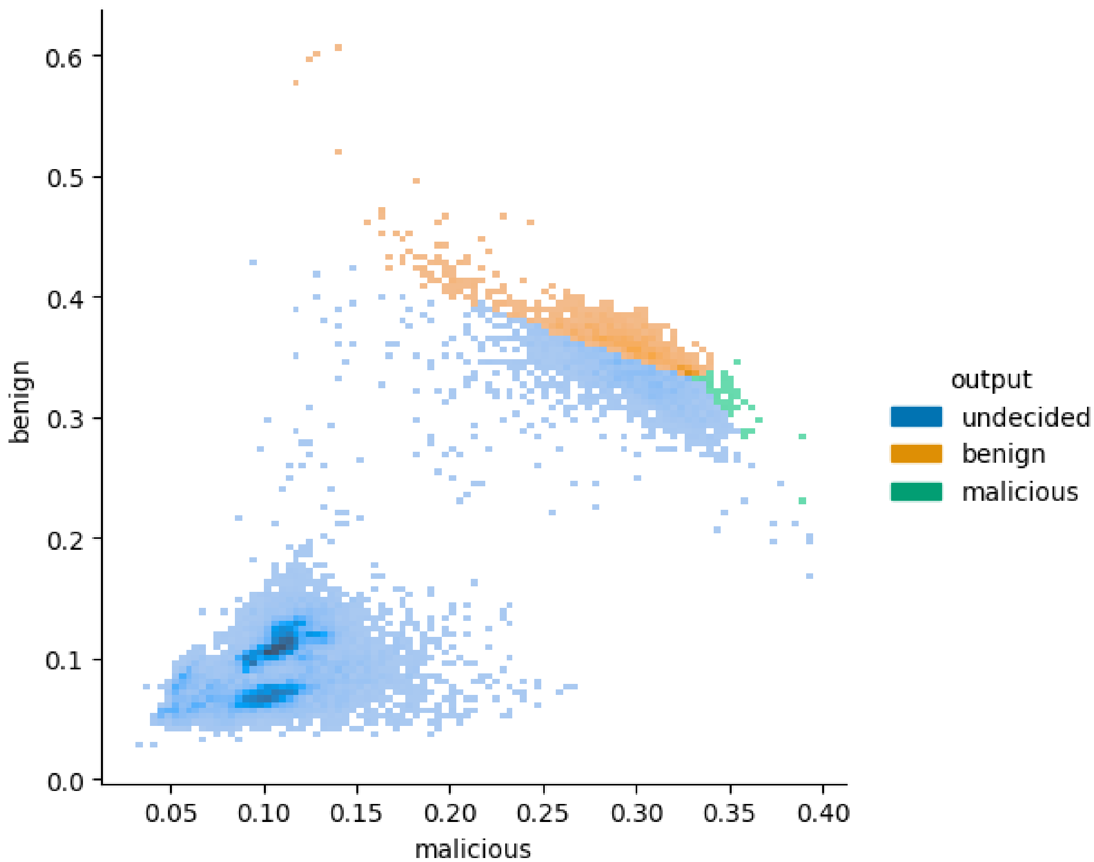
```
<seaborn.axisgrid.FacetGrid object at 0x7f9fe4165c30>
<seaborn.axisgrid.FacetGrid object at 0x7f9fe4165c30>
```



Somewhat, enlarged: (please, attention to the **axis**)

[45]:
```
sns_plot = sns.displot(
    data=df.select("malicious", "benign", "output").to_pandas(),
    x="malicious",
    y="benign",
        hue="output",
        palette=sns.color_palette("colorblind", n_colors=3)
)
print(sns_plot)
```

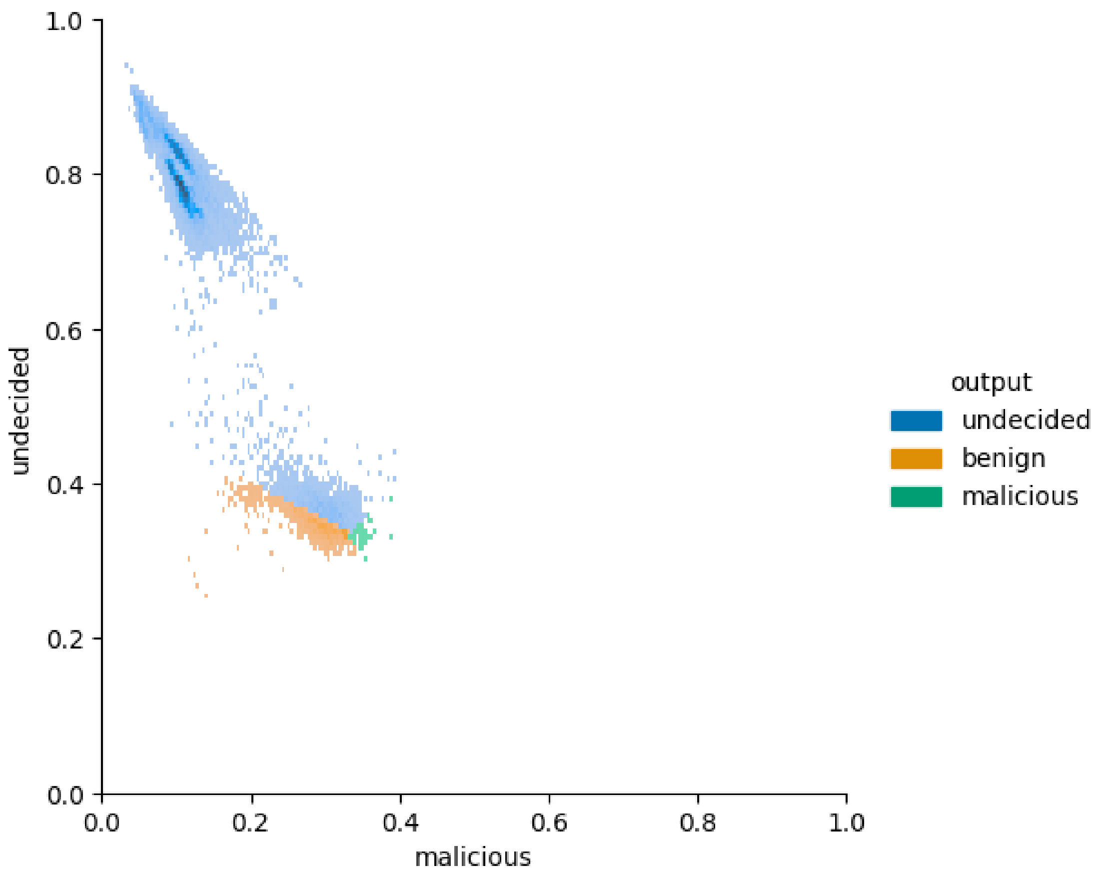
<seaborn.axisgrid.FacetGrid object at 0x7f9fe4137c40>



[51]:
```
sns_plot = sns.displot(
    data=df.select("undecided", "malicious", "output").to_pandas(),
    x="malicious",
    y="undecided",
        hue="output",
    palette=sns.color_palette("colorblind", n_colors=3)
)
```

```
sns_plot.set(ylim=(0,1), xlim=(0,1))
print(sns_plot)
```

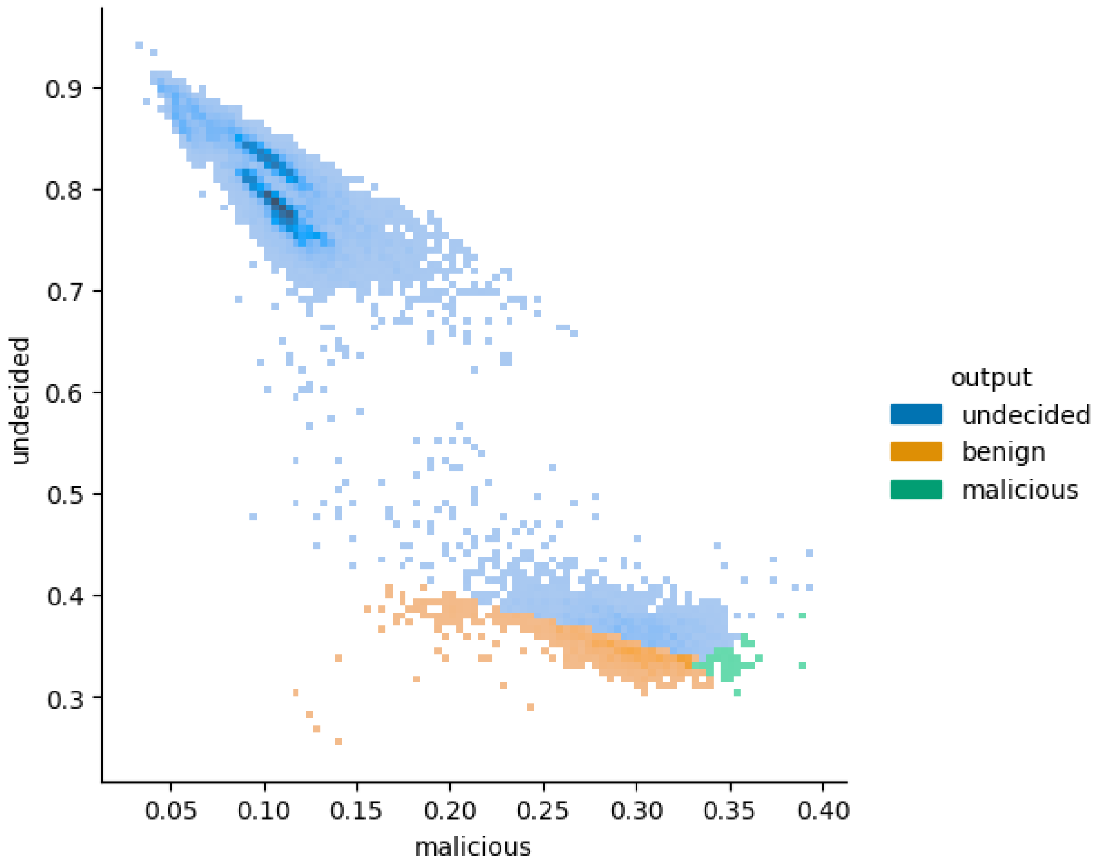
```
<seaborn.axisgrid.FacetGrid object at 0x7f9fe3f7bfa0>
```



Even this time, we can better see when we zoom in. Please, attention to the axis shift.

[50]:
```
sns_plot = sns.displot(
    data=df.select("undecided", "malicious", "output").to_pandas(),
    x="malicious",
    y="undecided",
        hue="output",
    palette=sns.color_palette("colorblind", n_colors=3)
)
print(sns_plot)
```

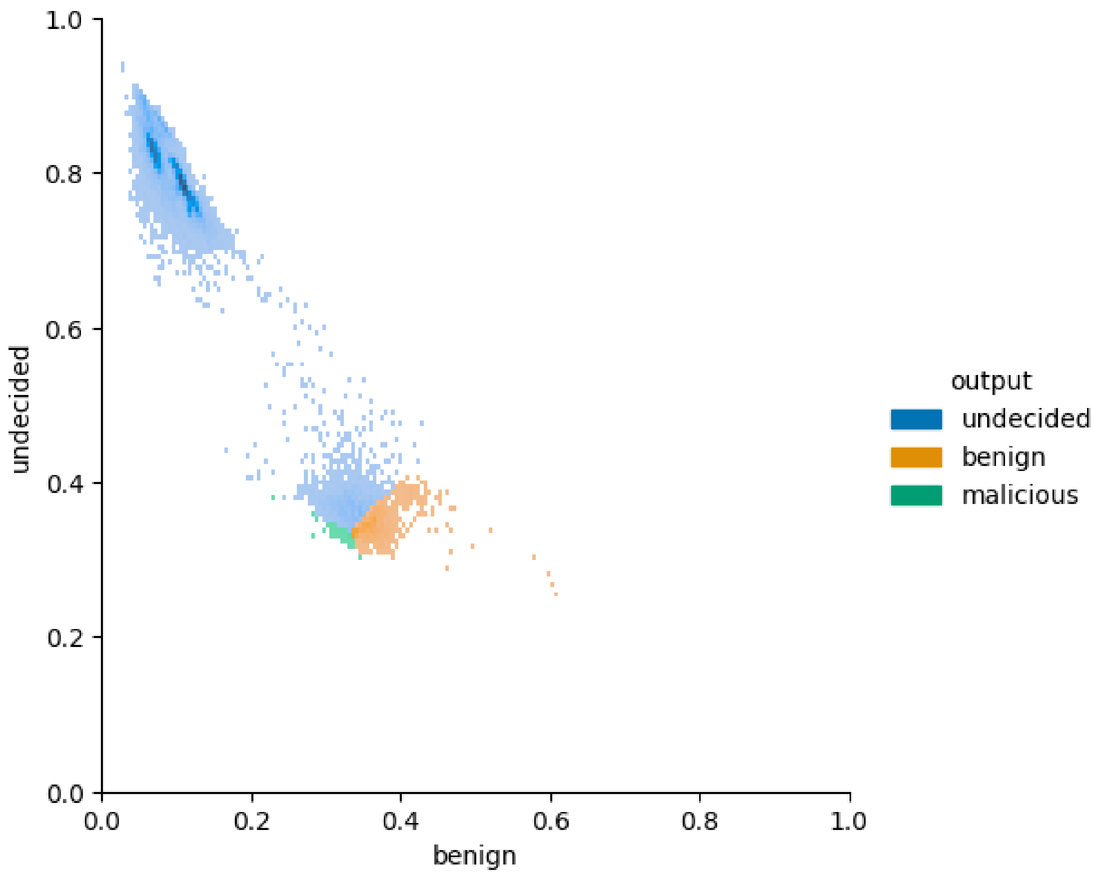
```
<seaborn.axisgrid.FacetGrid object at 0x7f9fe3e73d90>
```



And the last couple:

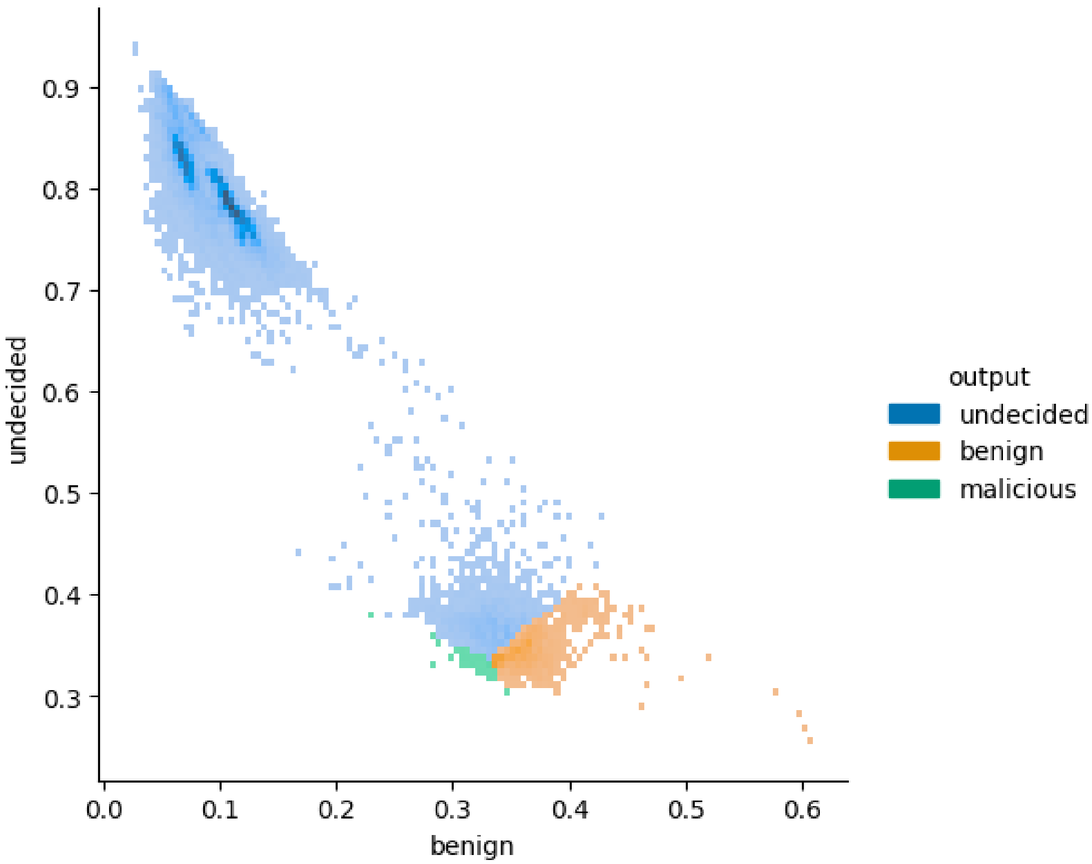
[53]:
```
sns_plot = sns.displot(
    data=df.select("undecided", "benign", "output").to_pandas(),
    x="benign",
    y="undecided",
        hue="output",
    palette=sns.color_palette("colorblind", n_colors=3)
)

sns_plot.set(ylim=(0,1), xlim=(0,1))
print(sns_plot)
```

```
<seaborn.axisgrid.FacetGrid object at 0x7f9fe4a33a60>
```



[54]:
```
sns_plot = sns.displot(
    data=df.select("undecided", "benign", "output").to_pandas(),
    x="benign",
    y="undecided",
        hue="output",
    palette=sns.color_palette("colorblind", n_colors=3)
)
print(sns_plot)
```

```
<seaborn.axisgrid.FacetGrid object at 0x7f9fe43efa00>
```



[55]:
```
# import plotly.express as px

# fig = px.scatter_3d(df.to_pandas(), x='malicious', y='benign', z='undecided',
  ↪color="output")
# fig.show()
# This is the 3D interactive mode, not abilitated when pdf.
```