Streamit URL

Connect-4 dataset

I chose a dataset that contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced.

There are 2 players (X and O), the outcome of the match is either win, lose or draw for player X. I chose this because I already know what connect-4 is and how it works so it will be easier to interpret and understand the data compared to a dataset that is about a topic where I barely know anything about.

What is an EDA?

Exploratory Data Analysis (EDA) is a method used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions. EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them

There are 8 steps in performing an EDA:

- Import Libraries and Load Dataset: importing necessary libraries such as pandas, numpy, graphviz, etc., and load your dataset.
- Check for Missing Values
- Visualizing the Missing Values: Use visual techniques to identify where the missing values are located
- Replacing the Missing Values: Decide on a strategy to handle these missing values
- Asking Analytical Questions and Visualizations: Formulate questions you want to answer from the dataset and use visualizations to find these answers
- Data Cleaning/Wrangling: Clean the data by removing duplicates, handling outliers, etc

- Feature Engineering: Explore various variables and their transformations to create new features or derive meaningful insights
- Statistics Summary: Generate summary statistics for numerical data in the dataset

Import Libraries and Load Dataset

The first step is to install and import the packages and load the dataset.

In [ ]:
```
!pip install graphviz
!pip install pydotplus
!pip install pandas
!pip install seaborn
!pip install streamlit
```

```
Requirement already satisfied: graphviz in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (0.20.1)
[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: pydotplus in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from pydotplus) (3.1.1)
[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: pandas in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (2.1.2)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from pandas) (1.26.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-pack
ages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fr
om pandas) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from pyt
hon-dateutil>=2.8.2->pandas) (1.16.0)
[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: seaborn in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (0.13.0)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packag
es (from seaborn) (1.26.1)
Requirement already satisfied: pandas>=1.2 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
seaborn) (2.1.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.3 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-pac
kages (from seaborn) (3.8.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from matplotlib!=3.6.1,>=3.3->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
matplotlib!=3.6.1,>=3.3->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from matplotlib!=3.6.1,>=3.3->seaborn) (4.44.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from matplotlib!=3.6.1,>=3.3->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (f
rom matplotlib!=3.6.1,>=3.3->seaborn) (23.1)
Requirement already satisfied: pillow>=8 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from ma
tplotlib!=3.6.1,>=3.3->seaborn) (10.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from matplotlib!=3.6.1,>=3.3->seaborn) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packag
es (from matplotlib!=3.6.1,>=3.3->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
pandas>=1.2->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fr
om pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from pyt
hon-dateutil>=2.7->matplotlib!=3.6.1,>=3.3->seaborn) (1.16.0)
[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: streamlit in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (1.28.1)
Requirement already satisfied: altair<6,>=4.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fr
om streamlit) (5.1.2)
Requirement already satisfied: blinker<2,>=1.0.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (1.7.0)
Requirement already satisfied: cachetools<6,>=4.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (5.3.2)
Requirement already satisfied: click<9,>=7.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fro
m streamlit) (8.1.7)
Requirement already satisfied: importlib-metadata<7,>=1.4 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-
packages (from streamlit) (6.8.0)
Requirement already satisfied: numpy<2,>=1.19.3 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (1.26.1)
Requirement already satisfied: packaging<24,>=16.8 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-package
s (from streamlit) (23.1)
Requirement already satisfied: pandas<3,>=1.3.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (2.1.2)
Requirement already satisfied: pillow<11,>=7.1.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (10.1.0)
Requirement already satisfied: protobuf<5,>=3.20 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (4.25.0)
Requirement already satisfied: pyarrow>=6.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
streamlit) (14.0.1)
Requirement already satisfied: python-dateutil<3,>=2.7.3 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-p
ackages (from streamlit) (2.8.2)
Requirement already satisfied: requests<3,>=2.27 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from strea
mlit) (2.31.0)
Requirement already satisfied: rich<14,>=10.14.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (13.6.0)
Requirement already satisfied: tenacity<9,>=8.1.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (8.2.3)
Requirement already satisfied: toml<2,>=0.10.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (f
rom streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site
-packages (from streamlit) (4.8.0)
Requirement already satisfied: tzlocal<6,>=1.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (f
rom streamlit) (5.2)
Requirement already satisfied: validators<1,>=0.2 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (0.22.0)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in c:\users\michiel\appdata\local\programs\python\python311\lib\sit
e-packages (from streamlit) (3.1.40)
```

```
Requirement already satisfied: pydeck<1,>=0.8.0b4 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (0.8.1b0)
Requirement already satisfied: tornado<7,>=6.0.3 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from streamlit) (6.3.3)
Requirement already satisfied: watchdog>=2.1.5 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (f
rom streamlit) (3.0.0)
Requirement already satisfied: jinja2 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from altair<6,>=4.0->
streamlit) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from altair<
6,>=4.0->streamlit) (4.19.1)
Requirement already satisfied: toolz in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from altair
<6,>=4.0->streamlit) (0.12.0)
Requirement already satisfied: colorama in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from cli
ck<9,>=7.0->streamlit) (0.4.6)
Requirement already satisfied: gitdb<5,>=4.0.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (f
rom gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.11)
Requirement already satisfied: zipp>=0.5 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from im
portlib-metadata<7,>=1.4->streamlit) (3.17.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
pandas<3,>=1.3.0->streamlit) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fr
om pandas<3,>=1.3.0->streamlit) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from pyt
hon-dateutil<3,>=2.7.3->streamlit) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\michiel\appdata\roaming\python\python311\site-packages (fro
m requests<3,>=2.27->streamlit) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from requests<
3,>=2.27->streamlit) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from requ
ests<3,>=2.27->streamlit) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from requ
ests<3,>=2.27->streamlit) (2023.7.22)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packa
ges (from rich<14,>=10.14.0->streamlit) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-pac
kages (from rich<14,>=10.14.0->streamlit) (2.16.1)
Requirement already satisfied: smmap<6,>=3.0.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (f
rom gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->streamlit) (5.0.1)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from jinja2-
>altair<6,>=4.0->streamlit) (2.1.3)
Requirement already satisfied: attrs>=22.2.0 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from jsonschem
```

```
a>=3.0->altair<6,>=4.0->streamlit) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\users\michiel\appdata\roaming\python\python311\site-p
ackages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from jso
nschema>=3.0->altair<6,>=4.0->streamlit) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\michiel\appdata\roaming\python\python311\site-packages (from jsonsche
ma>=3.0->altair<6,>=4.0->streamlit) (0.10.3)
Requirement already satisfied: mdurl~=0.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from m
arkdown-it-py>=2.2.0->rich<14,>=10.14.0->streamlit) (0.1.2)
[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (1.3.
2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from scikit-learn) (1.26.1)
Requirement already satisfied: scipy>=1.5.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fro
m scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packag
es (from scikit-learn) (3.2.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

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

The dataset is read and put in a dataframe.

In [ ]:
```
# load dataset
connect_df = pd.read_csv("connect-4.data/connect-4.data", sep=',')
```

I use the describe function to give a summary of the central tendency, dispersion, and shape of the distribution of a dataset, excluding NaN values. The describe function didn't show all the columns so I had to define this manually to use 43 columns because the dataset has 43 columns.

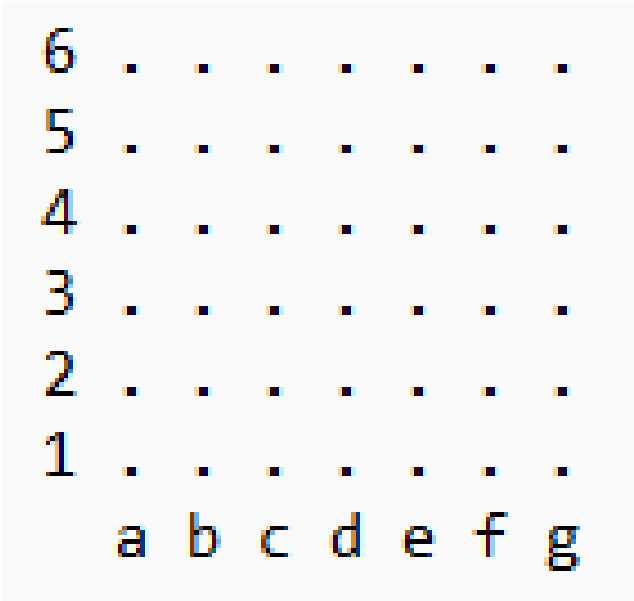
In [ ]:
```
with pd.option_context('display.max_columns', 43):
    print(connect_df.describe())
```

```
             b    b.1    b.2    b.3    b.4    b.5    b.6    b.7    b.8    b.9  \
count    67556  67556  67556  67556  67556  67556  67556  67556  67556  67556
unique       3      3      3      3      3      3      3      3      3      3
top          b      b      b      b      b      b      x      b      b      b
freq     24981  43384  55332  61615  65264  67039  25889  41179  54351  61205

          b.10   b.11      x      o   b.12   b.13   b.14   b.15    x.1    o.1  \
count    67556  67556  67556  67556  67556  67556  67556  67556  67556  67556
unique       3      3      3      3      3      3      3      3      3      3
top          b      b      x      b      b      b      b      b      b      b
freq     65203  67041  32032  37671  52897  60722  65142  67043  29016  46933

           x.2    o.2    x.3    o.3   b.16   b.17   b.18   b.19   b.20   b.21  \
count    67556  67556  67556  67556  67556  67556  67556  67556  67556  67556
unique       3      3      3      3      3      3      3      3      3      3
top          b      b      b      b      b      b      b      b      b      b
freq     57856  63096  66041  67245  40846  55526  62351  65449  66927  67472

          b.22   b.23   b.24   b.25   b.26   b.27   b.28   b.29   b.30   b.31  \
count    67556  67556  67556  67556  67556  67556  67556  67556  67556  67556
unique       3      3      3      3      3      3      3      3      3      3
top          b      b      b      b      b      b      b      b      b      b
freq     34166  51168  60373  64838  66818  67468  29728  48103  58868  64300

          b.32   b.33    win
count    67556  67556  67556
unique       3      3      3
top          b      b    win
freq     66709  67464  44472
```

Here I get my first problem where the dataset didn't contain column names. This is a problem because the describe function takes the top row to define the columns of the dataset. In this case the top row also has values:

[b,b,b,b,b,b,b,b,b,b,b,b,x,o,b,b,b,b,x,o,x,o,x,o,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,b,win] so I have to change the dataset to include column names.

The dataset was originally made with this layout for the connect-4 playboard where the possible spots are represented like cöordinates:



So I applied these cöordinates to the names of the columns in the updated dataset:

In [ ]:
```
# load updated dataset
updatedConnect_df = pd.read_csv("connect-4.data/connect-4-with-column-names.data", sep=',')

with pd.option_context('display.max_columns', 43):
    print(updatedConnect_df.describe())
```

```
           A1     A2     A3     A4     A5     A6     B1     B2     B3     B4  \
count   67557  67557  67557  67557  67557  67557  67557  67557  67557  67557
unique      3      3      3      3      3      3      3      3      3      3
top         b      b      b      b      b      b      x      b      b      b
freq    24982  43385  55333  61616  65265  67040  25889  41180  54352  61206

           B5     B6     C1     C2     C3     C4     C5     C6     D1     D2  \
count   67557  67557  67557  67557  67557  67557  67557  67557  67557  67557
unique      3      3      3      3      3      3      3      3      3      3
top         b      b      x      b      b      b      b      b      b      b
freq    65204  67042  32033  37671  52898  60723  65143  67044  29016  46933

           D3     D4     D5     D6     E1     E2     E3     E4     E5     E6  \
count   67557  67557  67557  67557  67557  67557  67557  67557  67557  67557
unique      3      3      3      3      3      3      3      3      3      3
top         b      b      b      b      b      b      b      b      b      b
freq    57856  63096  66041  67245  40847  55527  62352  65450  66928  67473

           F1     F2     F3     F4     F5     F6     G1     G2     G3     G4  \
count   67557  67557  67557  67557  67557  67557  67557  67557  67557  67557
unique      3      3      3      3      3      3      3      3      3      3
top         b      b      b      b      b      b      b      b      b      b
freq    34167  51169  60374  64839  66819  67469  29729  48104  58869  64301

           G5     G6 Outcome
count   67557  67557   67557
unique      3      3       3
top         b      b     win
freq    66710  67465   44473
```

As you can see I only get 4 statistics back because the dataset contains String values. the 4 statistics are describes as such:

- Count: Number of non-null (or, non-NaN) observations. This gives you the total number of non-null entries in the column.
  - In this case we have 67556 rows + 1 row for the column names for each column. This also confirms that there are no missing values because the amount of rows in the dataset is also 67556.
- Unique: Number of distinct values in a column. This tells you how many different categories or labels are present in the column.
  - There are 3 unique values for each possible cöordinate on the board:
    - b = blank
    - x = player x has taken this cöordinate

    - o = player o has taken this cöordinate
- Top: Most frequent value. This is the value that appears most often in the column.
    - for the most part the blank option is the most used in each row except for row A6 and B6.
- Freq: Frequency of the most frequent value. This tells you how many times the most frequent value appears in the column.

Show the data on the board

In this code you can see how a row in the dataset represents a state on the board.

In [ ]:
```
# Select a row from the DataFrame
selected_row = updatedConnect_df.iloc[10]  # You can change this to select other rows

# Now I convert the selected row into a 7x6 numpy array
board = np.array(selected_row[:-1]).reshape(7, 6)

# Transpose the board
board = board.T
# Reverse the rows
board = board[::-1]

# Print the board
print(board)
```

```
[['b' 'b' 'b' 'o' 'b' 'b' 'b']
 ['b' 'b' 'b' 'x' 'b' 'b' 'b']
 ['b' 'b' 'b' 'o' 'b' 'b' 'b']
 ['b' 'b' 'b' 'x' 'b' 'b' 'b']
 ['b' 'b' 'b' 'o' 'b' 'b' 'b']
 ['o' 'x' 'b' 'x' 'b' 'b' 'b']]
```

Split the data

Given the cöordinates I will predict if the outcome will result in a win a loss or a draw. N?ow to split the features and the target variable:

In [ ]:
```
# split dataset in features and target variable
```

```
feature_cols = ['A1','A2','A3','A4','A5','A6','B1','B2','B3','B4','B5','B6','C1','C2','C3','C4','C5','C6','D1','D2','D3','D4',

X = updatedConnect_df[feature_cols]
y = updatedConnect_df[['Outcome']] # target variable
```

In [ ]:
```
print(X)
```

```
      A1 A2 A3 A4 A5 A6 B1 B2 B3 B4  ... F3 F4 F5 F6 G1 G2 G3 G4 G5 G6
0      b  b  b  b  b  b  b  b  b  b  ...  b  b  b  b  b  b  b  b  b  b
1      b  b  b  b  b  b  b  b  b  b  ...  b  b  b  b  b  b  b  b  b  b
2      b  b  b  b  b  b  o  b  b  b  ...  b  b  b  b  b  b  b  b  b  b
3      b  b  b  b  b  b  b  b  b  b  ...  b  b  b  b  b  b  b  b  b  b
4      o  b  b  b  b  b  b  b  b  b  ...  b  b  b  b  b  b  b  b  b  b
...   .. .. .. .. .. .. .. .. .. ..  ... .. .. .. .. .. .. .. .. .. ..
67552  x  x  b  b  b  b  o  x  o  b  ...  b  b  b  b  o  o  x  b  b  b
67553  x  x  b  b  b  b  o  b  b  b  ...  b  b  b  b  o  x  o  o  x  b
67554  x  x  b  b  b  b  o  o  b  b  ...  b  b  b  b  o  x  x  o  b  b
67555  x  o  b  b  b  b  o  b  b  b  ...  b  b  b  b  o  x  o  x  x  b
67556  x  o  o  o  x  b  o  b  b  b  ...  b  b  b  b  x  b  b  b  b  b

[67557 rows x 42 columns]
```

In [ ]:
```
print(y)
```

```
      Outcome
0         win
1         win
2         win
3         win
4         win
...       ...
67552    loss
67553    draw
67554    loss
67555    draw
67556    draw

[67557 rows x 1 columns]
```

Firstly I need to train the machine learning techniques. To do this I first need to install `category_encoders` to replace the categories with numeric values because **the decision trees implemented in scikit-learn use only numerical features and these features are interpreted**

always as continuous numeric variables.

In [ ]:
```
pip install --upgrade category_encoders
```

```
Requirement already satisfied: category_encoders in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(2.6.3)
Requirement already satisfied: numpy>=1.14.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fro
m category_encoders) (1.26.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packag
es (from category_encoders) (1.3.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
category_encoders) (1.11.3)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages
(from category_encoders) (0.14.0)
Requirement already satisfied: pandas>=1.0.5 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fro
m category_encoders) (2.1.2)
Requirement already satisfied: patsy>=0.5.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
category_encoders) (0.5.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-pack
ages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from
pandas>=1.0.5->category_encoders) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fr
om pandas>=1.0.5->category_encoders) (2023.3)
Requirement already satisfied: six in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (from patsy>=
0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (fro
m scikit-learn>=0.20.0->category_encoders) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packag
es (from scikit-learn>=0.20.0->category_encoders) (3.2.0)
Requirement already satisfied: packaging>=21.3 in c:\users\michiel\appdata\local\programs\python\python311\lib\site-packages (f
rom statsmodels>=0.9.0->category_encoders) (23.1)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Here I replace the categories with numerical values using `One-Hot encoding`. With one-hot encoding, we convert each categorical value into a new categorical value and assign a binary value of 1 or 0.

In [ ]:
```
import category_encoders as ce

ce_oh = ce.OneHotEncoder(cols = feature_cols)
X_cat_oh = ce_oh.fit_transform(X)
```

What the encoder does is put a 1 on the spot where it sees something is 'filled in'. every feature is splitted into 3 columns because there are 3 possible values for each feature. A1 becomes [A1_1, A1_2, A1_3], these 3 values represent the possible values [b, o, x] and if for example b is filled in in A1 then A1_1 gets the value 1.

In [ ]:
```
print(X_cat_oh)
```

```
       A1_1  A1_2  A1_3  A2_1  A2_2  A2_3  A3_1  A3_2  A3_3  A4_1  ...  G3_3  \
0         1     0     0     1     0     0     1     0     0     1  ...     0
1         1     0     0     1     0     0     1     0     0     1  ...     0
2         1     0     0     1     0     0     1     0     0     1  ...     0
3         1     0     0     1     0     0     1     0     0     1  ...     0
4         0     1     0     1     0     0     1     0     0     1  ...     0
...     ...   ...   ...   ...   ...   ...   ...   ...   ...   ...  ...   ...
67552     0     0     1     0     0     1     1     0     0     1  ...     1
67553     0     0     1     0     0     1     1     0     0     1  ...     0
67554     0     0     1     0     0     1     1     0     0     1  ...     1
67555     0     0     1     0     1     0     1     0     0     1  ...     0
67556     0     0     1     0     1     0     0     1     0     0  ...     0

       G4_1  G4_2  G4_3  G5_1  G5_2  G5_3  G6_1  G6_2  G6_3
0         1     0     0     1     0     0     1     0     0
1         1     0     0     1     0     0     1     0     0
2         1     0     0     1     0     0     1     0     0
3         1     0     0     1     0     0     1     0     0
4         1     0     0     1     0     0     1     0     0
...     ...   ...   ...   ...   ...   ...   ...   ...   ...
67552     1     0     0     1     0     0     1     0     0
67553     0     1     0     0     0     1     1     0     0
67554     0     1     0     1     0     0     1     0     0
67555     0     0     1     0     0     1     1     0     0
67556     1     0     0     1     0     0     1     0     0

[67557 rows x 126 columns]
```

The dataset still needs to be splitted into training data and testing data. The process of splitting the data involves randomly assigning data points to either the training set or the test set. A common split ratio is 80% for training and 20% for testing, but this can vary depending on the size and nature of a dataset.

Here I split 20% of the data into test data and 80% into training data. I also use a seed for the random number generator used for the split.

In [ ]:
```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_cat_oh, y, test_size=0.2, random_state=42)
```

These are the labels for the confusion matrix

In [ ]:
```
labels = ['draw', 'lose', 'win']
```

Decision Tree

For the first ML learning algorithm I will use the decision tree.

A decision tree is capable of performing classification on a dataset. It takes 2 arrays as input:

- Array X, these are the training samples so in this case the positions players have taken on the board.
- Array Y, these are the labels for the training samples so in this case the outcome of the sample (win, lose, draw).

Now we can try to train the classifier (decision tree) with the training data.

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(criterion = "entropy")
clf = clf.fit(X_train, y_train)
```

Now we can make predictions using the test data.

In [ ]:
```
predictions = clf.predict(X_test)
```

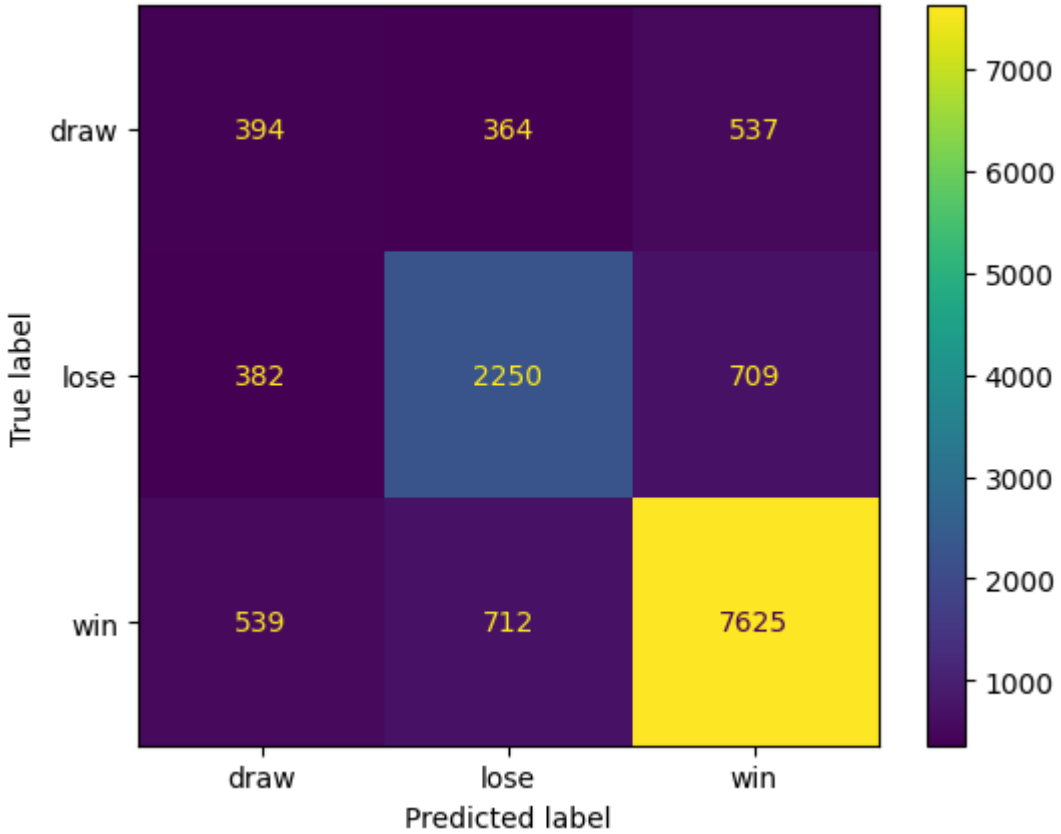
And then calculate the accuracy of the algorithm using the actual target values of the test data and compare them with the predicted values.

I also make a confusion matrix to get an overview of the predictions.

In [ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: ", round(accuracy, 2) * 100, "%")

# compute the confusion matrix
cmTree = confusion_matrix(y_test, predictions)
ConfusionMatrixDisplay(cmTree, display_labels=labels).plot()
```

```
Accuracy:  76.0 %
```

Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1c58d4fb3d0>
```



In the graph above you can see when the model predicted right and when it went wrong.

The labels on the left are what the outcome of the test data actually was whilst the labels on the bottom are the predictions the model made.

So for example with the decision tree if a sample had an outcome 'draw' the model would predict the right outcome 391 times (top left square).

The other times it would predict 'draw' 382 times for an outcome that was supposed to be 'lose'.

From this confusion matrix we can see that it got a lot of predictions right when it came to the outcome 'win', this is because there are a lot more samples in the dataset where the actual outcome is 'win' so it was trained better to predict that outcome.

Support Vector Machine

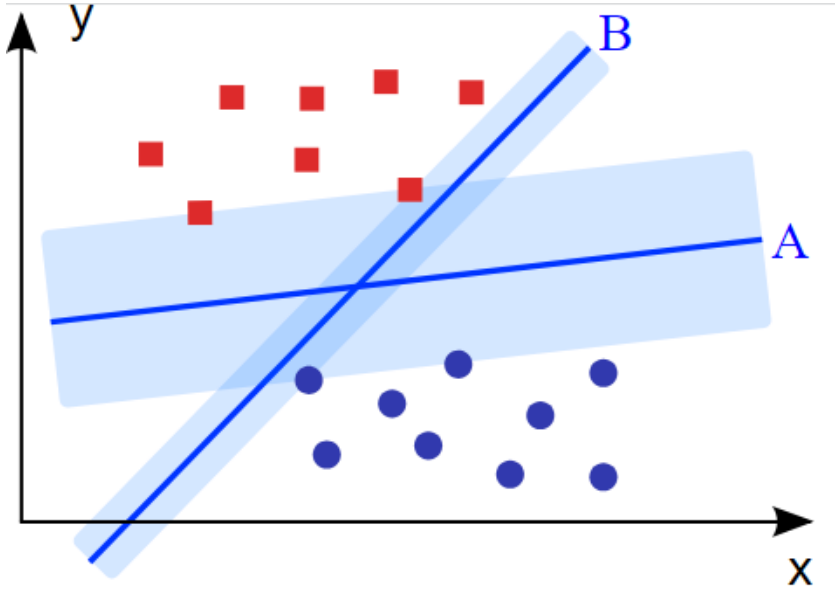
Support Vector Machine (SVM) is a machine learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks, but it is best suited for classification. SVMs are adaptable and efficient in a variety of applications because they can manage high-dimensional data.

What the SVM algorithm does is find the optimal **hyperplane** in a N-dimensional space that separates the data in different classes. In other words the distance between the closest data points from 2 different classes should be as big as possible. The hyperplane serves as a decision boundary to seperate all the data points of different classes in the feature space.

The dimension of the hyperplane depends on the amount of features in the dataset, here it will be a 42-dimensional hyperplane.

How does it work with an example

In a dataset with 2 features the hyperplane is a line because 2 features means it is 2 dimensional. Let's say we have 2 classes in the dataset represented as blue and red in the image:



The algorithm then chooses which hyperplane is located the furthest away from both the nearest data points from the red and blue classes. This hyperplane is also known as the **maximum-margin hyperplane**.

In the image above the best **maximum-margin hyperplane** is clearly "A" because the distance from the 2 nearest data points from both red and blue is greater then the distance from the 2 nearest data points from both red and blue on hyperplane "B".

Outliers

What if a red data point is located under the hyperplane (so in the area of the blue data points)?

In this case the maximum-margin hyperplane is located the same way as normal, but a penalty is given each time a data point crosses the hyperplane. so here a red point in the blue area gives a penalty.

Extra

This technique takes about 5 minutes to train and test on this dataset in the code below as well as in streamlit.

In [ ]:
```
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
```

I Create a StandardScaler instance to normalize the feature values

The I Fit the scaler to the training data and transform it. I also do this with the test data.

In [ ]:
```
# Create a scaler
scaler = StandardScaler()

# Fit the scaler to the training data and transform
X_train_scaled = scaler.fit_transform(X_train)

# Transform the test data
X_test_scaled = scaler.transform(X_test)
```

I Create a Support Vector Machine (SVM) classifier

I Train the SVM classifier on the scaled training data

It Makes predictions on the scaled test data

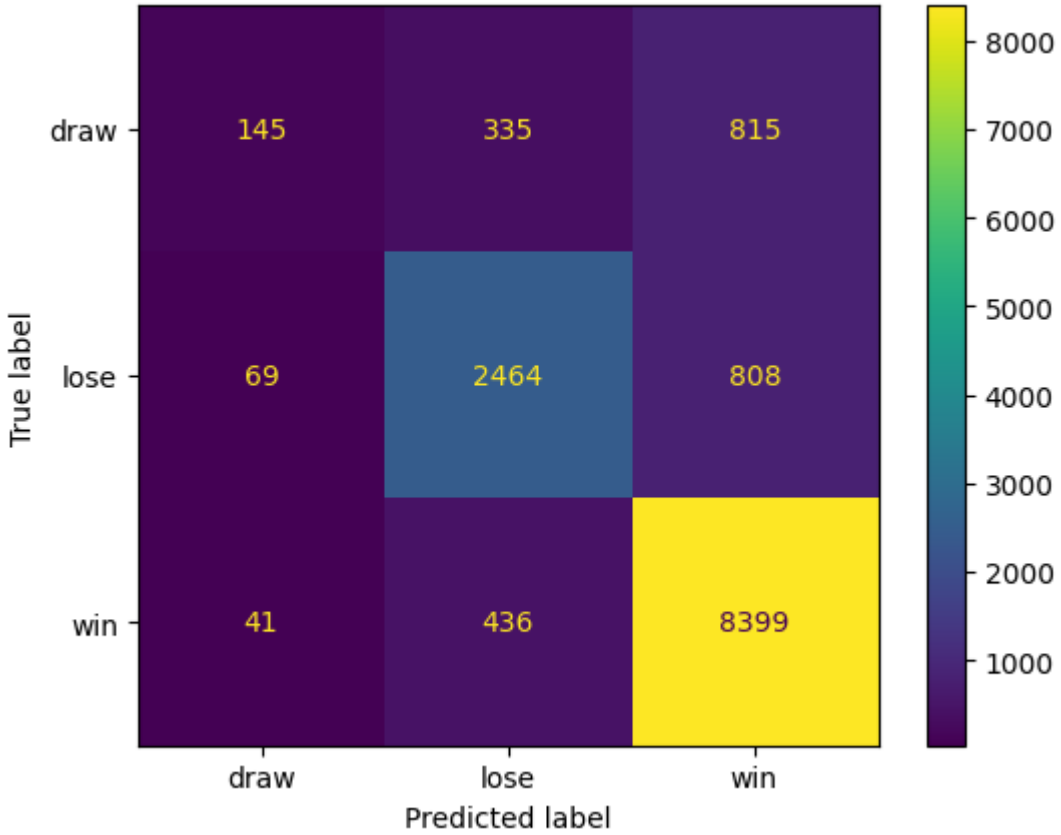
In [ ]:
```
svm_clf = SVC()
svm_clf.fit(X_train_scaled, np.ravel(y_train))

predictions = svm_clf.predict(X_test_scaled)
```

Here I calculate the accuracy of the model by comparing predicted labels with true labels and I plot a confusion matrix to evaluate the model's performance.

In [ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: ", round(accuracy, 2) * 100, "%")

# compute the confusion matrix
cmVector = confusion_matrix(y_test, predictions)
ConfusionMatrixDisplay(cmVector, display_labels=labels).plot()
```

```
Accuracy:  81.0 %
```

Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1c58f848110>
```



Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent algorithm that is used for optimizing machine learning models. It is more efficient when dealing with larger datasets compared to the regular Gradient Descent algorithm.

In SGD, instead of using the entire dataset for each iteration, only a single random training example or batch is selected to calculate the gradient and update the model parameters.

How does it work

there are 4 steps in the algorithm:

- Initialization: first the parameters of the model are randomly initialized.
- Set parameters: then the parameters are set by defining the number of iterations and the learning rate.
- Stochastic Gradient descent loop: here there are 5 more steps:
  - The training dataset is shuffled to achieve randomness
  - Then we iterate over each training sample or batch
  - Compute the gradient of the cost function with respect to the model parameters using the current training example (or batch).
  - Update the model parameters by taking a step in the direction of the negative gradient, scaled by the learning rate.
  - Evaluate the convergence criteria, such as the difference in the cost function between iterations of the gradient.
- Return optimized parameters: Once the convergence criteria are met or the maximum number of iterations is reached, return the optimized model parameters.

Here I Create an instance of the SGD Classifier with specified parameters

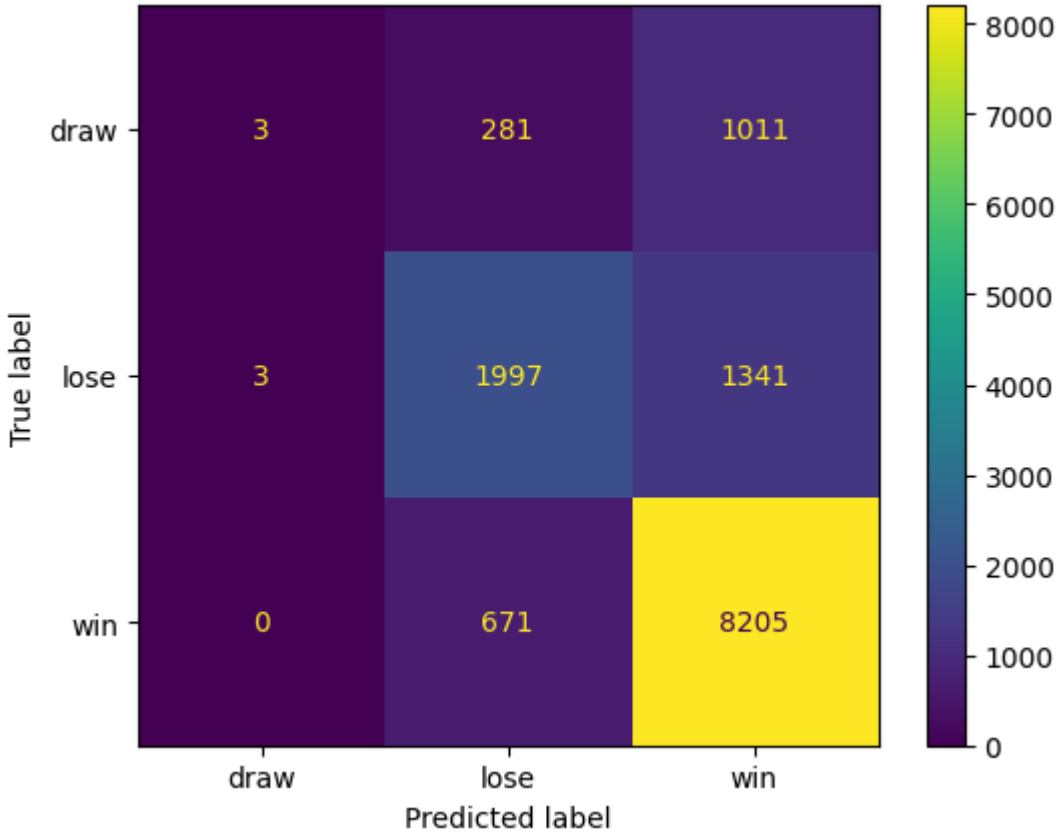
In [ ]:
```
from sklearn.linear_model import SGDClassifier

lr_clf = SGDClassifier(loss='hinge', penalty='l2', max_iter=100)
lr_clf.fit(X_train, np.ravel(y_train))

predictions = lr_clf.predict(X_test)
```

Here I calculate the accuracy of the model by comparing predicted labels with true labels and I plot a confusion matrix to evaluate the model's performance.

In [ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, predictions)
print("Accuracy: ", round(accuracy, 2) * 100, "%")

# compute the confusion matrix
cmGradient = confusion_matrix(y_test, predictions)
ConfusionMatrixDisplay(cmGradient, display_labels=labels).plot()
```

```
Accuracy:  76.0 %
```

Out[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1c58f8c1bd0>



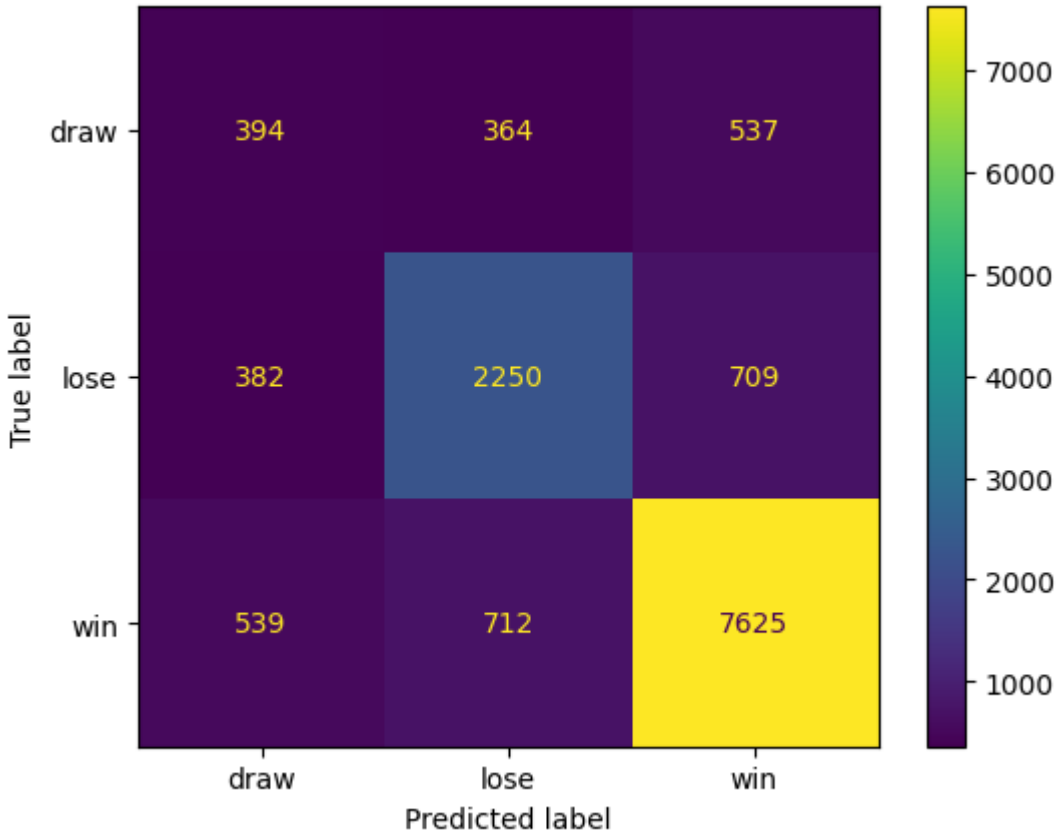
Comparison

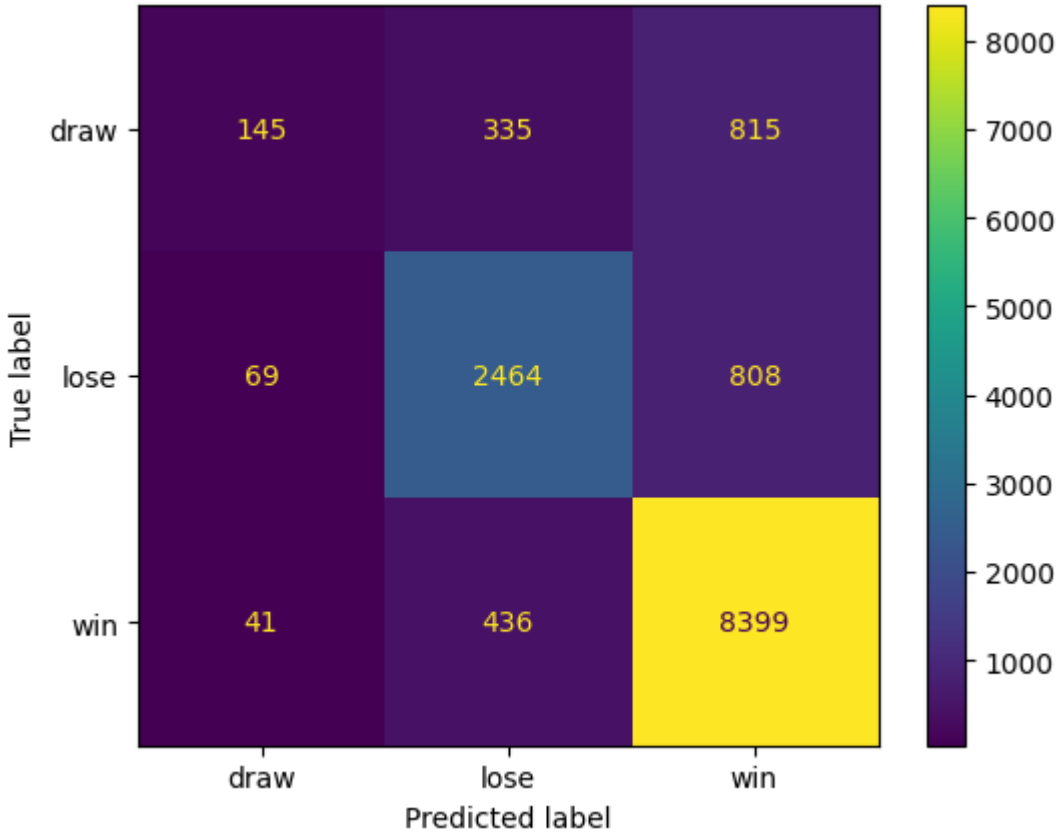
In [ ]:
```
print("Decision Tree")
ConfusionMatrixDisplay(cmTree, display_labels=labels).plot()

print("Support Vector Machine")
ConfusionMatrixDisplay(cmVector, display_labels=labels).plot()
```

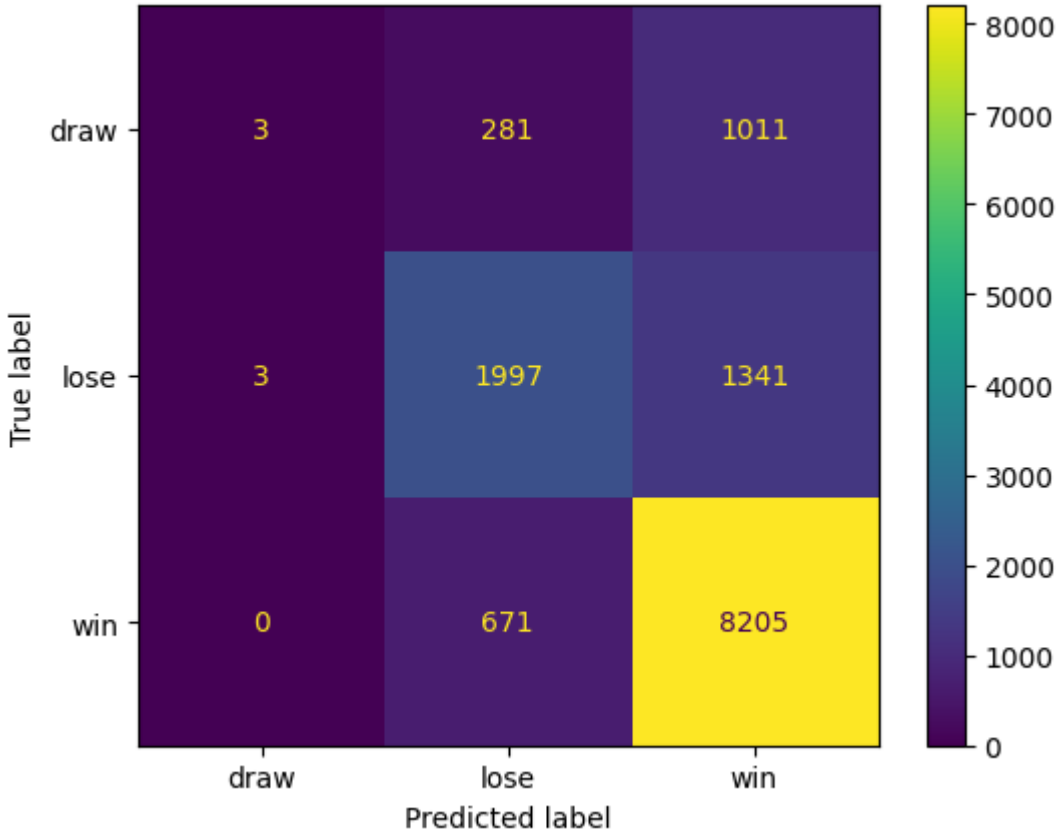
```
print("Stochastic Gradient Descent")
ConfusionMatrixDisplay(cmGradient, display_labels=labels).plot()
```

```
Decision Tree
Support Vector Machine
Stochastic Gradient Descent
```

Out[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1c58fb59e90>




145
335
815
69
2464
808
41
436
8399
draw
lose
win
True label
draw
lose
win
Predicted label
8000
7000
6000
5000
4000
3000
2000
1000



To compare the different algorithms we can look at their confusion matrices.

- Top confusion matrix: Decision Tree
- Middle confusion matrix: Support Vector Machine
- Bottom confusion matrix: Stochastic Gradient Descent

In each confusion matrix the correct prediction for each of the possible outcomes is shown in the diagonal from the top left corner to the bottom right corner. The other values are predictions that were not correct. With this in mind we can see which algorithm is used best for this dataset.

What immediatly catches my eye is that the Stochastic Gradient descent algorithm doesn't work well if the label from the dataset has more than 2 options. I can conlude this from the matrix because it barely predicts any draws whilst the other algorithms predict a decent amount of

draws. This makes the SGD the worst option to use on this dataset.

The decision tree makes a lot of false predictions when it predicts it is a draw. This is probably due to the draw outcome being the least common in the training dataset. The SVM algorithm is better at predicting the right outcomes for all possibilities. This makes it the most accurate of the 3 algorithms but it also takes about 5 minutes to train, so if a faster solution is needed a decision tree is best to use else the SVM gives more accurate predictions.

Generative AI Tools

Prompts used

- What is an EDA and how do I perform this on a dataset? - BingAI - with this prompt a very long explanation was given about EDA followed by the steps that need to be performed in that process.
- Using the pandas library explain what describe does - BingAI - I get an explanation of all the statistics from the describe function output and the function is also shown iin an example but the statistics aren't the same as mine because i have a dataset with String values so I had to ask further:
    - I apply this function on a dataset with string values so explain the statistics from that result - bingAI - Now it gives me the right statisticsand explains them.
- Briefly explain the ML learning alogrithm for a decision tree - BingAI - This gives a short explanation of the algorithm with an example, in this example it also shows how to divide the dataset into training data and test data so I asked more information about this:
    - explain how the splitting of the data into training data and test data works - BingAI - Here it gives a reason why data is splitted into test and training data and it also says that usually 80% of the data is training data and 20% is test data.
- Explain what one hot encoding is - BingAI - Here it gives a complicated explanation of what one-hot encoding is.
- show how I can make an accuracy of the decision tree when i'm using one hot encoding - BingAI - Here it gives example code showing how I can measure the accuracy of the algorithm, but it only showed how to get the accuracy so I asked if there were more measurements possible:
    - are there more ways of measuring the ML algorithm so i can compare it to other algorityhms? - BingAI - It then gives different ways for measuring ML algorithms for classification, regression, clustering and ranking.