Name: Apoorva Singh
Group (hours):

Introduction to Artificial Intelligence, 2022/2023
Supervised Learning – Report

Tasks:
https://colab.research.google.com/drive/1pbLae5QGH24Bd-LuyGkXwMoFxrT-thtK

My copies:
https://colab.research.google.com/drive/1HeQOMThxcV6Lpi1gOdh7KnAMRJ2eBY_h
(alic@772g) (tasks)

https://colab.research.google.com/drive/16M8i7hmUWkpZl4aVKNsvIlSK2XCpQXn7?authuser=1#scrollTo=nNhlNbdci7BA
(asingh@g) (q and a)

How to:
Create a copy of the notebook (line above) and work on your account.
If there are any problems (errors, sth is unclear) with the code/task – put them using the comment function in the file provided above or report to the teacher.

Part 1:

1. **[0] What are the differences between supervised, unsupervised, and reinforcement learning?**

   Supervised- labeled data is given to predict, the result is adjusted by a human moderator every so often to make sure the output is what is desired.
   eg. labeled tom and jerry

   Unsupervised- unlabeled data is given to be categorized, this process is not supervised and the grouping is left to the AI.
   eg. unlabeled tom and jerry

   Reinforcement- The actor receives reward for a desired output and punishment for an undesired one.
   eg. pigeon superstition.

2. **[0] How is regression different from classification?**

   Regression- helps predict a continuous quantity or values such as price, income, age, etc. Regression predictions can be evaluated using root mean squared error.

   Classification- predicts discrete class labels and are used to forecast or classify the distinct values such as Real or False, Male or Female, Spam or Not Spam,
   eg. Golf prediction program.
   Classification predictions can be evaluated using accuracy.

3. **[0] What are linear regression, decision tree, and neural network?**

Linear Regression-

- is used to determine the strength of predictors
- to forecast an effect
- and to forecast a trend

It is a type of predictive analysis used to explain the relationship between one dependent variable and one or more independent variables.

Decision tree- A Decision tree is the denotative representation of a decision-making process. In the presentation the decision tree was illustrated with the example of if it being optimal to play golf on that day or not. The decision was found to be independent of being sunny or cloudy, but rain prompted a negative decision. This model is based on supervised learning, in specific contexts based on the available data.

Neural Network- Are based on the human brain dendrite structure, comprising of nodes. It is divided into input, output and hidden layers. Every node is given a weight and threshold, when activated it passes data along to the next layer this is why its known as a feedforward network.. The nodes can be thought of as their own linear regression model. The challenge mentioned here was to find the optimal weight for tasks.

4. **[0] What is the test set, and what is the training set?**

Test set- is used to confirm that the AI learned correctly from the unseen data from the training set. 80% of the data should be testing.

Training set- trains a model, it is designed to set the basis for situations and decisions, this is the data set that the AI is first given, it is usually unseen. 20% should be training.

5. **[0] What is *overfitting*, and what is *underfitting*?**

Underfitting- happens when the AI cannot capture the underlying trend of the data, it performs well on training data but the testing data result is too loose. It implies that the rules are too relaxed and flexible to produce accurate results from a smaller data set.

Overfitting- happens when the AI cannot capture the underlying trend of the data, it performs well on training data but the testing data result is too loose. It implies that the rules are too relaxed and flexible to produce accurate results from a smaller or high variance data set.

6. **[0] How to deal with *overfitting* for linear regression, decision trees, and neural networks?**

   Overfitting can be dealt with by training the algorithm with **more relevant data, augmenting** the data sets. Alternatively, one can also partition the data into subsets by training with **cross-validation**. Being more mindful of the **features** that are needed, **selecting** them or prioritizing them as such, guiding the algorithm better is another solution along with **regularizationL (Lasso, Ridge, Elastic Net).**

7. **[0] What is Exploratory Data Analysis (EDA), and why do we do it?**

   Originally developed by American mathematician John Tukey in the 1970s, EDA is used to analyze and investigate data sets, summarize their main characteristic and represent the same using data visualization methods. It is used to **discover patterns, spot anomalies, test a hypothesis,** or **check assumptions** by showing how to best manipulate the data.

   Techniques to perform EDA include: Clustering and dimension reduction techniques, **Univariate, Bivariate and Multivariate visualizations, K-means Clustering, and Predictive models.**

8. **[0] What is an *outlier*?**
   An outlier is a data point on a graph or in a set of results that is very much bigger or smaller than the next nearest data point, it throws off the curve and does not fit into the normal distribution, outliers are generally undesirable **anomalies or errors**, they do however help in **providing the confidence** and can occasionally give interesting insights though are largely undesirable.

9. **[0] What metrics can be used to measure the model's performance in supervised learning, and what is their interpretation? Name at least four.**

   Accuracy = no of correct predictions / no of total predictions
   this is the most intuitive model.

   Confusion Matrix Accuracy = no of correct predictions / np of total predictions
   = (TP + TN) / (TP + TN + FP + FN)
   results are put into a matrix *where t= true p=positive n= negative*

   F1 Score
   Precision = TP / (TP + FP)
   Recall = Sensitivity = TPR = TP/ (TP + FN)
   based on confusion matrix

   Mean Absolute Error

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

compute the average of the absolute value of residuals.

10. [0] What is cross-validation?

In order to compare and select an appropriate model for a specific machine learning predictive modeling problem, we have to use an evaluation and performance testing technique, one such would be cross-validation, it can be simplified into 4 steps:

1. Dividing the dataset into two parts: training and testing
2. Training the model by training set
3. Validating the model by test set
4. Repeating steps 1 2 3 a couple of times, depending on the cross validation method used.

.

Part 2: no questions

Am aware work was not necessary to show but i still wanted to:

```
[10] def Fibonacci(n):
         if n < 0: print("Incorrect input")
         elif n == 0: return 0
         elif n == 1 or n == 2: return 1
         else: return Fibonacci(n-1) + Fibonacci(n-2)
     i= 12 #i being the nth place in fib
     fibarr=[] # initializing an array

     for j in range(i): fibarr+=[Fibonacci(j)]
     print("the output for", i ," digits of the Fibonacci array is ", fibarr)

     the output for 12  digits of the Fibonacci array is  [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

partial program check (we covered this same program in highschool and i vividly rember it coming in my 10th grade final exam, i wanted to make sure it was the correct code)

https://www.geeksforgeeks.org/python-program-for-program-for-fibonacci-numbers-2/

after checking the solution was indeed as i recalled, i modified the code to generate the positions in an array and return an array with for loop. code can be optimized more with less lines (the if loops can be really compressed in python to scary extents sometimes) but i wanted to make it more readable at the moment :)

Part 3: no questions

Part 4:

1. [1] What is the size (columns, rows) of the dataset?

```
[6] from google.colab import files

    uploaded = files.upload() # load the file with teh swimming dataset
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving swimming.csv to swimming (1).csv
```

```
[42] import pandas as pd
     import numpy as np
     import matplotlib
     import matplotlib.pyplot as plt
     import seaborn as sns
     import missingno
     %matplotlib inline
```

```
[35] df = pd.read_csv('swimming.csv') # displays rows and columns of dataset
     print(df.shape)
```

```
(6, 7)
```

```
[33] df.info() # displays rows and columns of dataset and other info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   sky       6 non-null      object
 1   airTemp   6 non-null      object
 2   humidity  6 non-null      object
 3   wind      6 non-null      object
 4   water     6 non-null      object
 5   forecast  6 non-null      object
 6   enjoy     6 non-null      object
dtypes: object(7)
memory usage: 464.0+ bytes
```

```
[34] df.dtypes
```

```
sky         object
airTemp     object
humidity    object
wind        object
water       object
forecast    object
enjoy       object
dtype: object
```

2. **[1] How many attributes are there in the dataset? Which of them are the feature variables? What are the others? given is a list of the variables, there are 6 attributes and the feature variables are with higher relevance.**

```python
def uniq(df):
    columns = df.select_dtypes(include=['object', 'category']).columns
    for i in columns:
        print(" ")
        print("---------------------")
        print("Unique variables of " + i)
        print("---------------------")
        print(df[i].value_counts().reset_index().rename(columns={"index": i, i: "n"})[:min(3, len(df[i].value_counts()))])
        print(" ")
uniq(df)
```

```
---------------------
Unique variables of sky
---------------------
      sky  n
0   sunny  3
1  cloudy  2
2   rainy  1
```

```
---------------------
Unique variables of airTemp
---------------------
  airTemp  n
0    warm  4
1    cold  2


---------------------
Unique variables of humidity
---------------------
  humidity  n
0     high  4
1   normal  2


---------------------
Unique variables of wind
---------------------
     wind  n
0  strong  4
1    weak  2


---------------------
Unique variables of water
---------------------
  water  n
0  warm  4
1  cool  2
```

```
--------------------
Unique variables of forecast
--------------------
  forecast  n
0     same  4
1   change  2


--------------------
Unique variables of enjoy
--------------------
  enjoy  n
0   yes  4
1    no  2
```

3. [1] How many instances are there? How many positive (enjoy == yes) and negative instances are in the dataset?
   also visible from the above code, there are 4 instances of yes and 2 of no

```
[32] swimming = df[df['enjoy'] != 'yes'] # choose only these rows, which represent 'yes' (by enjoy)
     swimming.head()
```

	sky	airTemp	humidity	wind	water	forecast	enjoy
2	rainy	cold	high	strong	warm	change	no
5	cloudy	cold	high	weak	cool	same	no

```
[36] swimming = df[df['humidity'] == 'high'] # choose only these rows, which represent 'yes' (by enjoy)
     swimming.head()
```

	sky	airTemp	humidity	wind	water	forecast	enjoy
1	sunny	warm	high	strong	warm	same	yes
2	rainy	cold	high	strong	warm	change	no
3	sunny	warm	high	strong	cool	change	yes
5	cloudy	cold	high	weak	cool	same	no

Using example with iris dataset answer the following questions.

4. [1] Which of the feature variables best separates the data in terms of *enjoy*?
   evidently "yes" best separated the data

```
df['enjoy'].value_counts()

yes    4
no     2
Name: enjoy, dtype: int64
```

5. [1] How many items in the dataset the humidity variable is set to high (humidity == high)? What are their numbers in the dataset (counting from 0)? **the humidity is set to high 4 times**

```
[70] swimming = df[df['humidity'] == 'high'] # choose only these rows, which represent 'yes' (by enjoy)
     swimming.head()
```

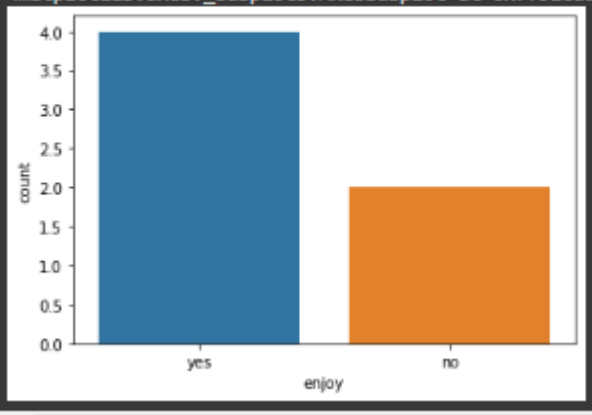
	sky	airTemp	humidity	wind	water	forecast	enjoy
1	sunny	warm	high	strong	warm	same	yes
2	rainy	cold	high	strong	warm	change	no
3	sunny	warm	high	strong	cool	change	yes
5	cloudy	cold	high	weak	cool	same	no

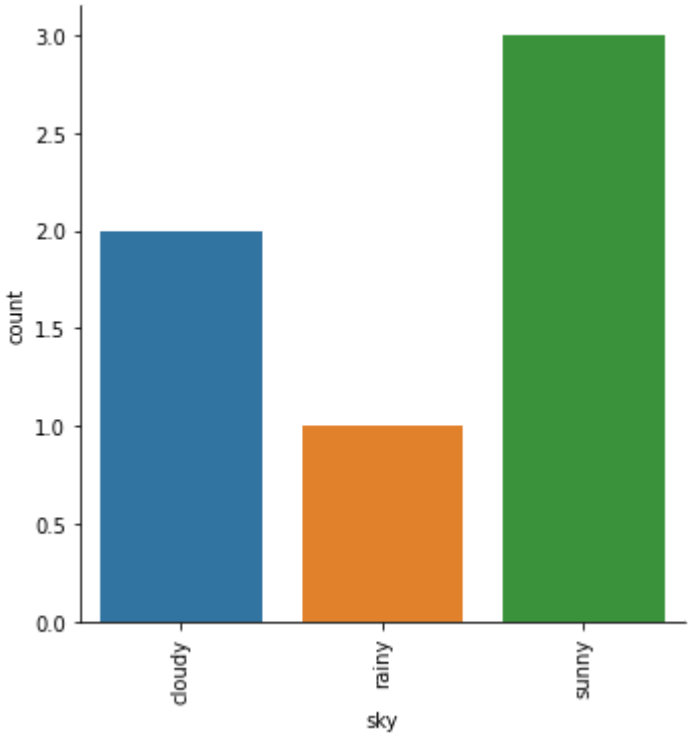
```
df['humidity'].value_counts()
```

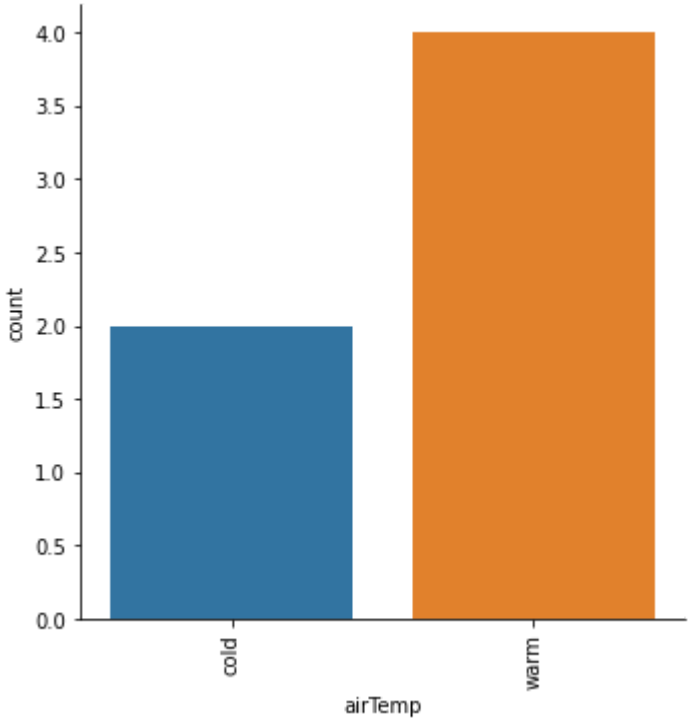
```
high      4
normal    2
Name: humidity, dtype: int64
```

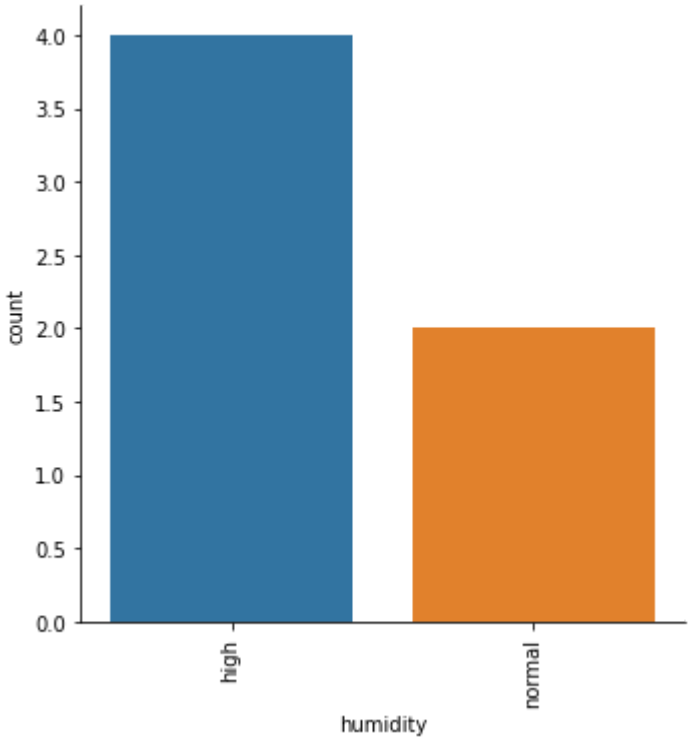
```
sns.countplot(df['enjoy'])
```

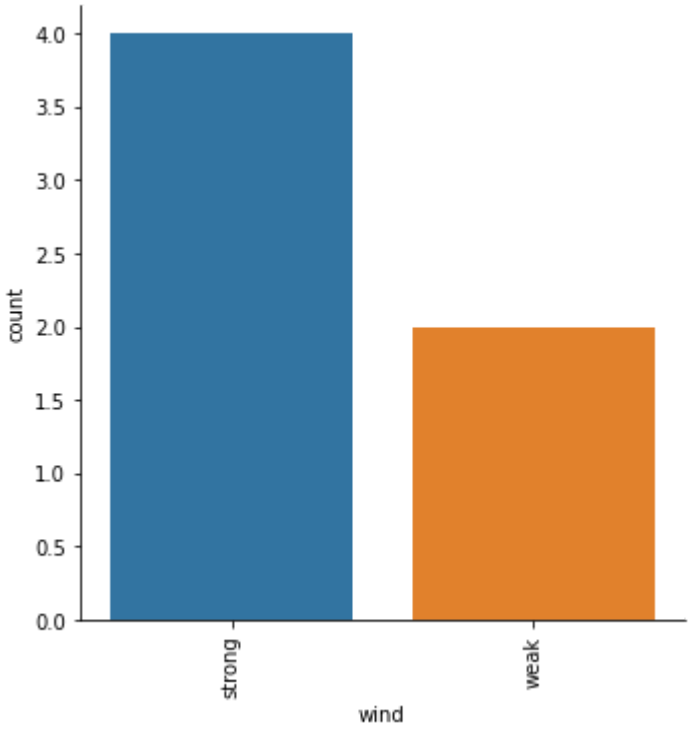
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:3
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f91cd9c9970>
```

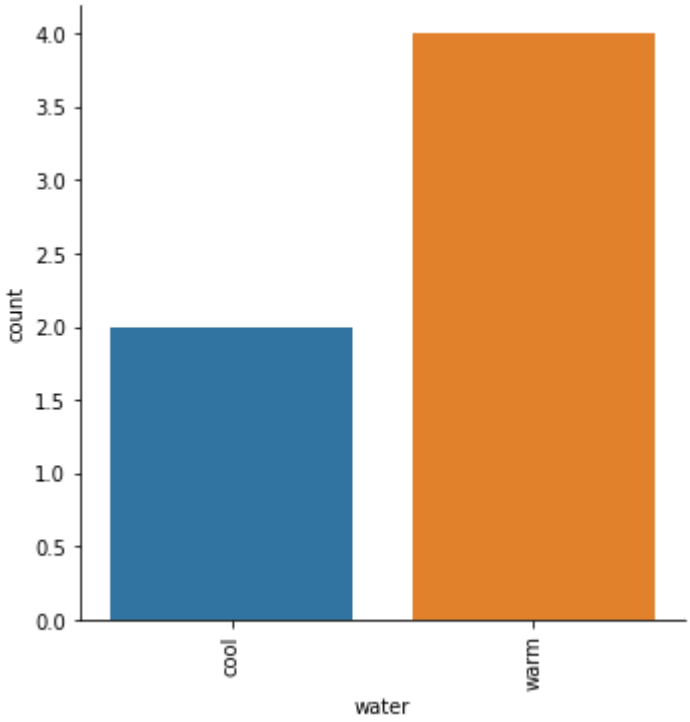


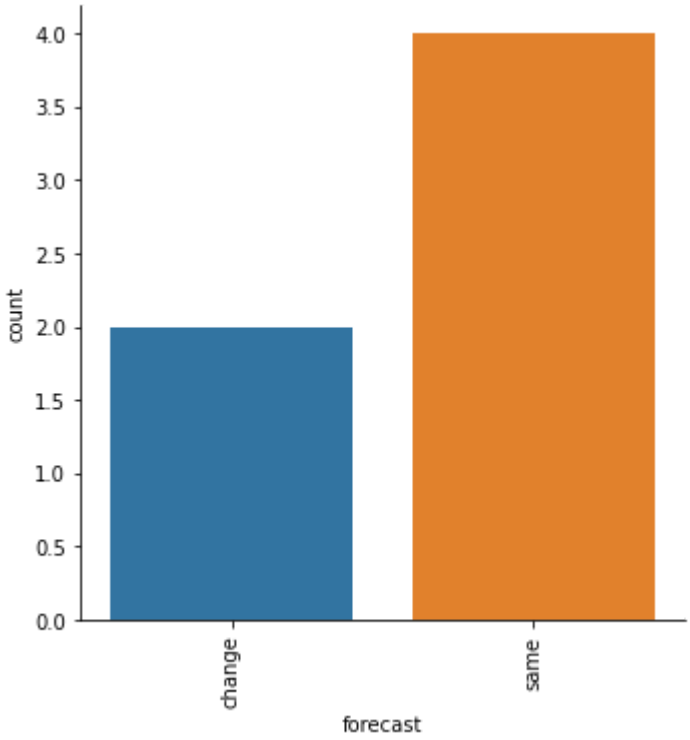

3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
cloudy
rainy
sunny
sky

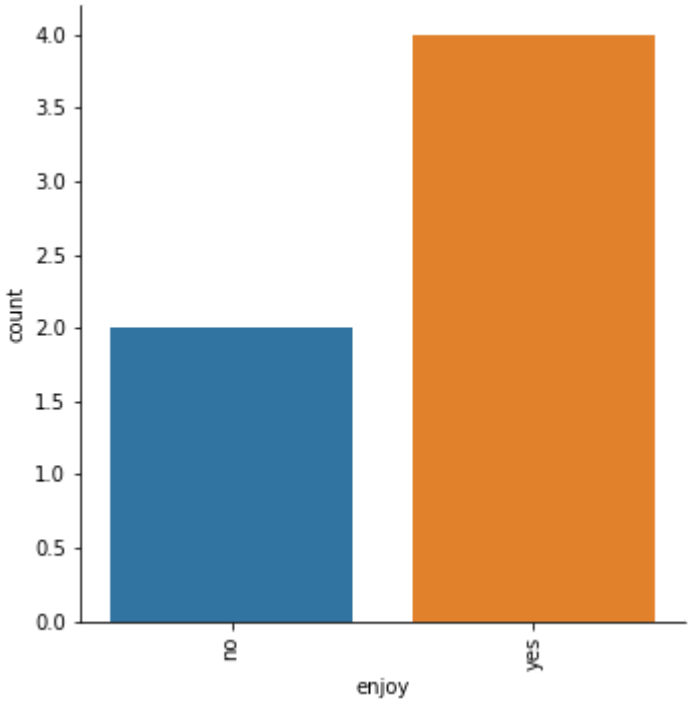

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
cold
warm
airTemp


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
high
normal
humidity


4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
strong
weak
wind

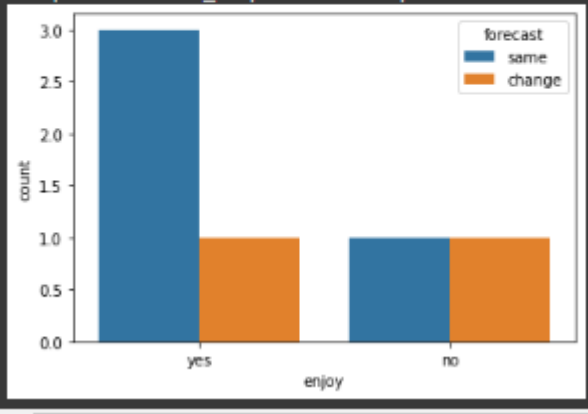

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
cool
warm
water

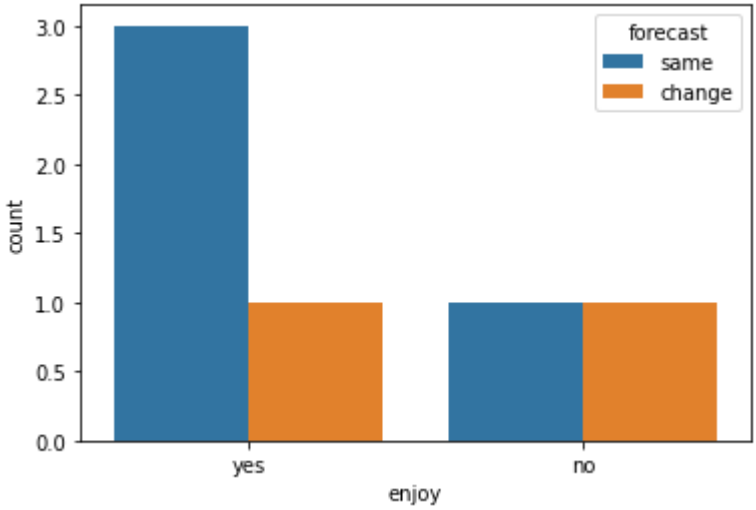

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
change
same
forecast

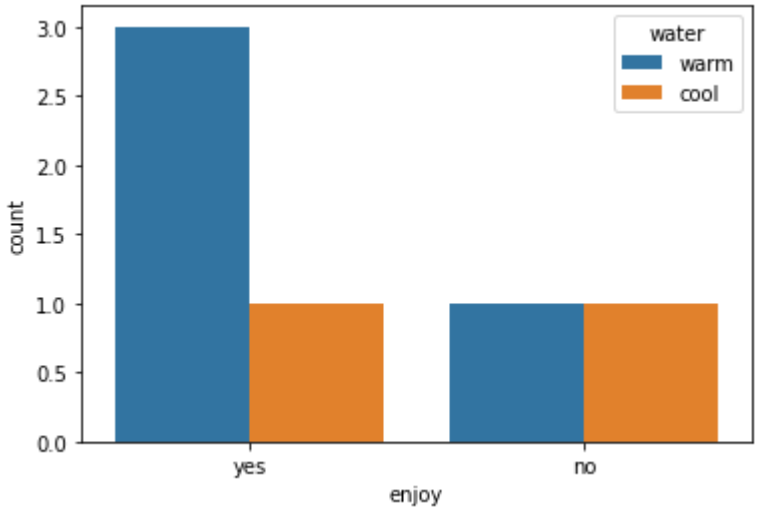

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
no
yes
enjoy

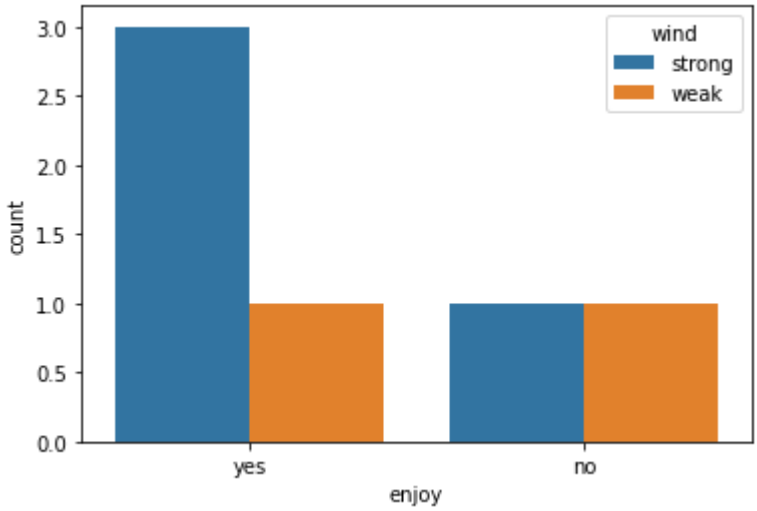
```
sns.countplot(df['enjoy'],hue=df['forecast'])
```

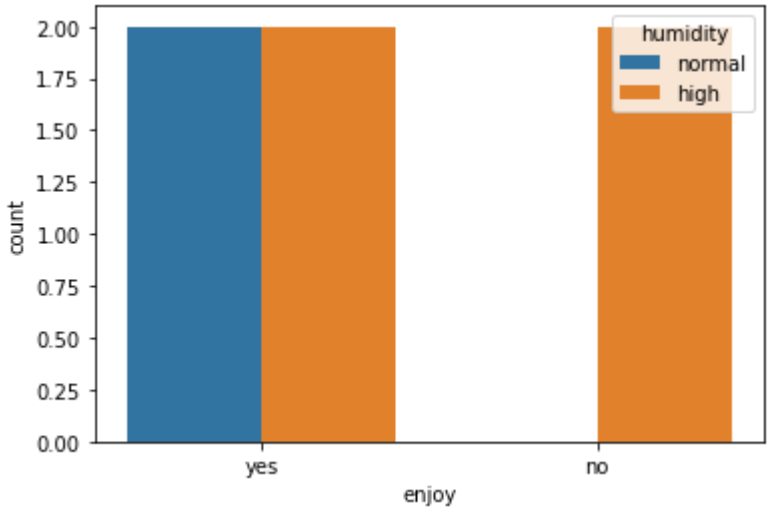
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f91df88e610>
```

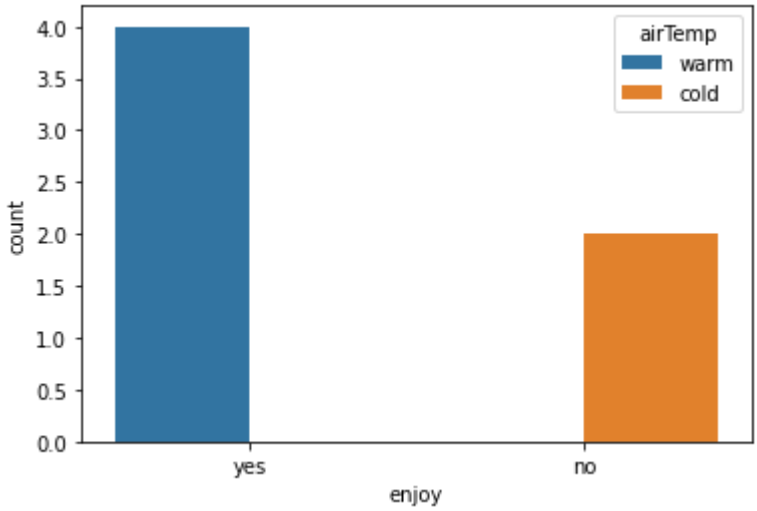





water
warm
cool
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
yes
no
enjoy


wind
strong
weak
3.0
2.5
2.0
1.5
1.0
0.5
0.0
count
yes
no
enjoy





Part 5:

1. **[1] How do these datasets differ?**

   they differ both in size and layout, df1 is cpu, df2 here is vendor cpu. the latter has the extra column

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209 entries, 0 to 208
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   MYCT    209 non-null    int64
 1   MMIN    209 non-null    int64
 2   MMAX    209 non-null    int64
 3   CACH    209 non-null    int64
 4   CHMIN   209 non-null    int64
 5   CHMAX   209 non-null    int64
 6   class   209 non-null    int64
dtypes: int64(7)
memory usage: 11.6 KB
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209 entries, 0 to 208
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   vendor  209 non-null    object
 1   MYCT    209 non-null    int64
 2   MMIN    209 non-null    int64
 3   MMAX    209 non-null    int64
 4   CACH    209 non-null    int64
 5   CHMIN   209 non-null    int64
 6   CHMAX   209 non-null    int64
 7   class   209 non-null    int64
dtypes: int64(7), object(1)
memory usage: 13.2+ KB
```

2. **[1] What type is the *vendor* data serie? vendor is a category type, listed as object**

```
[89] df1.dtypes

MYCT      int64
MMIN      int64
MMAX      int64
CACH      int64
CHMIN     int64
CHMAX     int64
class     int64
dtype: object
```

```
df2.dtypes

vendor    object
MYCT       int64
MMIN       int64
MMAX       int64
CACH       int64
CHMIN      int64
CHMAX      int64
class      int64
dtype: object
```

3. **[1] What happened after using the pd.get_dumies() function? Use hint!**

```
df2['vendor'] = pd.get_dummies(df2['vendor'])

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.8/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

                                4 frames
pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'vendor'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.8/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: 'vendor'

SEARCH STACK OVERFLOW
```

Without the hint: Code shows KeyError

Using the hint:The code executes but there is no output, it is just marked as correct.

```
# df = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset
#df['vendor'] = pd.get_dummies(df['vendor'])

df2 = pd.concat([pd.get_dummies(df2['vendor']), df2.drop('vendor', axis=1)], axis=1)
```

printing output and comparing to previous table
The vendor object type (categorical variable) has been converted into to dummy variables - - numeric variables that are used to represent categorical data.

```
df2 = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset
print(df2)
```

```
      vendor  MYCT  MMIN   MMAX  CACH  CHMIN  CHMAX  class
0    adviser   125   256   6000   256     16    128    199
1     amdahl    29  8000  32000    32      8     32    253
2     amdahl    29  8000  32000    32      8     32    253
3     amdahl    29  8000  32000    32      8     32    253
4     amdahl    29  8000  16000    32      8     16    132
..       ...   ...   ...    ...   ...    ...    ...    ...
204   sperry   124  1000   8000     0      1      8     37
205   sperry    98  1000   8000    32      2      8     50
206   sratus   125  2000   8000     0      2     14     41
207     wang   480   512   8000    32      0      0     47
208     wang   480  1000   4000     0      0      0     25

[209 rows x 8 columns]
```

```
# df = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset
#df['vendor'] = pd.get_dummies(df['vendor'])

df2 = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset

df2 = pd.concat([pd.get_dummies(df2['vendor']), df2.drop('vendor', axis=1)], axis=1)

print(df2)
```

```
     adviser  amdahl  apollo  basf  bti  burroughs  c.r.d  cambex  cdc  dec  \
0          1       0       0     0    0          0      0       0    0    0
1          0       1       0     0    0          0      0       0    0    0
2          0       1       0     0    0          0      0       0    0    0
3          0       1       0     0    0          0      0       0    0    0
4          0       1       0     0    0          0      0       0    0    0
..       ...     ...     ...   ...  ...        ...    ...     ...  ...  ...
204        0       0       0     0    0          0      0       0    0    0
205        0       0       0     0    0          0      0       0    0    0
206        0       0       0     0    0          0      0       0    0    0
207        0       0       0     0    0          0      0       0    0    0
208        0       0       0     0    0          0      0       0    0    0

     ...  sperry  sratus  wang  MYCT  MMIN   MMAX  CACH  CHMIN  CHMAX  class
0    ...       0       0     0   125   256   6000   256     16    128    199
1    ...       0       0     0    29  8000  32000    32      8     32    253
2    ...       0       0     0    29  8000  32000    32      8     32    253
3    ...       0       0     0    29  8000  32000    32      8     32    253
4    ...       0       0     0    29  8000  16000    32      8     16    132
..   ...     ...     ...   ...   ...   ...    ...   ...    ...    ...    ...
204  ...       1       0     0   124  1000   8000     0      1      8     37
205  ...       1       0     0    98  1000   8000    32      2      8     50
206  ...       0       1     0   125  2000   8000     0      2     14     41
207  ...       0       0     1   480   512   8000    32      0      0     47
208  ...       0       0     1   480  1000   4000     0      0      0     25

[209 rows x 37 columns]
```

4. **[1] Put the code for linear regression – from the moment the data are loaded.**

```
[1] import pandas as pd
    import numpy as np
    import matplotlib
    import matplotlib.pyplot as plt
    import seaborn as sns
    import missingno
    %matplotlib inline
    from sklearn.linear_model import LinearRegression
```

```
from google.colab import files

uploaded = files.upload() # load the file with teh swimming dataset
```

```
Choose Files  cpu.csv
• cpu.csv(text/csv) - 5071 bytes, last modified: 11/27/2022 - 100% done
Saving cpu.csv to cpu (3).csv
```

```
[3] from google.colab import files

    uploaded = files.upload() # load the file with teh swimming dataset
```

```
Choose Files  cpu_vendor.csv
• cpu_vendor.csv(text/csv) - 6330 bytes, last modified: 11/27/2022 - 100% done
Saving cpu_vendor.csv to cpu_vendor (2).csv
```

```
[4] df1 = pd.read_csv('cpu.csv') # displays rows and columns of dataset
    print(df1.shape)
```

```
(209, 7)
```

```
[5] df2 = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset
    print(df2.shape)
```

```
(209, 8)
```

```
[6] df1.to_numpy()
```

```
array([[  125,   256,  6000, ...,    16,   128,   198],
       [   29,  8000, 32000, ...,     8,    32,   269],
       [   29,  8000, 32000, ...,     8,    32,   220],
       ...,
       [  125,  2000,  8000, ...,     2,    14,    52],
       [  480,   512,  8000, ...,     0,     0,    67],
       [  480,  1000,  4000, ...,     0,     0,    45]])
```

```
[7] df2.to_numpy()
```

```
array([['adviser', 125, 256, ..., 16, 128, 199],
       ['amdahl', 29, 8000, ..., 8, 32, 253],
       ['amdahl', 29, 8000, ..., 8, 32, 253],
       ...,
       ['sratus', 125, 2000, ..., 2, 14, 41],
       ['wang', 480, 512, ..., 0, 0, 47],
       ['wang', 480, 1000, ..., 0, 0, 25]], dtype=object)
```

```
[8] df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209 entries, 0 to 208
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   MYCT    209 non-null    int64
 1   MMIN    209 non-null    int64
 2   MMAX    209 non-null    int64
 3   CACH    209 non-null    int64
 4   CHMIN   209 non-null    int64
 5   CHMAX   209 non-null    int64
 6   class   209 non-null    int64
dtypes: int64(7)
memory usage: 11.6 KB
```

```
df2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209 entries, 0 to 208
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   vendor  209 non-null    object
 1   MYCT    209 non-null    int64
 2   MMIN    209 non-null    int64
 3   MMAX    209 non-null    int64
 4   CACH    209 non-null    int64
 5   CHMIN   209 non-null    int64
 6   CHMAX   209 non-null    int64
 7   class   209 non-null    int64
dtypes: int64(7), object(1)
memory usage: 13.2+ KB
```

```
[10] df1.dtypes

MYCT     int64
MMIN     int64
MMAX     int64
CACH     int64
CHMIN    int64
CHMAX    int64
class    int64
dtype: object
```

```
df2.dtypes

vendor    object
MYCT       int64
MMIN       int64
MMAX       int64
CACH       int64
CHMIN      int64
CHMAX      int64
class      int64
dtype: object
```

```
[13] df2 = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset
     print(df2)
```

```
       vendor  MYCT  MMIN   MMAX  CACH  CHMIN  CHMAX  class
0     adviser   125   256   6000   256     16    128    199
1      amdahl    29  8000  32000    32      8     32    253
2      amdahl    29  8000  32000    32      8     32    253
3      amdahl    29  8000  32000    32      8     32    253
4      amdahl    29  8000  16000    32      8     16    132
..        ...   ...   ...    ...   ...    ...    ...    ...
204   sperry   124  1000   8000     0      1      8     37
205   sperry    98  1000   8000    32      2      8     50
206   sratus   125  2000   8000     0      2     14     41
207     wang   480   512   8000    32      0      0     47
208     wang   480  1000   4000     0      0      0     25

[209 rows x 8 columns]
```

```
# df = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset
#df['vendor'] = pd.get_dummies(df['vendor'])

df2 = pd.read_csv('cpu_vendor.csv') # displays rows and columns of dataset

df2 = pd.concat([pd.get_dummies(df2['vendor']), df2.drop('vendor', axis=1)], axis=

print(df2)
```

```
     adviser  amdahl  apollo  basf  bti  burroughs  c.r.d  cambex  cdc  dec  \
0          1       0       0     0    0          0      0       0    0    0
1          0       1       0     0    0          0      0       0    0    0
2          0       1       0     0    0          0      0       0    0    0
3          0       1       0     0    0          0      0       0    0    0
4          0       1       0     0    0          0      0       0    0    0
..       ...     ...     ...   ...  ...        ...    ...     ...  ...  ...
204        0       0       0     0    0          0      0       0    0    0
205        0       0       0     0    0          0      0       0    0    0
206        0       0       0     0    0          0      0       0    0    0
207        0       0       0     0    0          0      0       0    0    0
208        0       0       0     0    0          0      0       0    0    0

     ...  sperry  sratus  wang  MYCT  MMIN   MMAX  CACH  CHMIN  CHMAX  class
0    ...       0       0     0   125   256   6000   256     16    128    199
1    ...       0       0     0    29  8000  32000    32      8     32    253
2    ...       0       0     0    29  8000  32000    32      8     32    253
3    ...       0       0     0    29  8000  32000    32      8     32    253
4    ...       0       0     0    29  8000  16000    32      8     16    132
..   ...     ...     ...   ...   ...   ...    ...   ...    ...    ...    ...
204  ...       1       0     0   124  1000   8000     0      1      8     37
205  ...       1       0     0    98  1000   8000    32      2      8     50
206  ...       0       1     0   125  2000   8000     0      2     14     41
207  ...       0       0     1   480   512   8000    32      0      0     47
208  ...       0       0     1   480  1000   4000     0      0      0     25

[209 rows x 37 columns]
```

```
dataset.keys()
```

```
Index(['adviser', 'amdahl', 'apollo', 'basf', 'bti', 'burroughs', 'c.r.d',
       'cambex', 'cdc', 'dec', 'dg', 'formation', 'four-phase', 'gould',
       'harris', 'honeywell', 'hp', 'ibm', 'ipl', 'magnuson', 'microdata',
       'nas', 'ncr', 'nixdorf', 'perkin-elmer', 'prime', 'siemens', 'sperry',
       'sratus', 'wang', 'MYCT', 'MMIN', 'MMAX', 'CACH', 'CHMIN', 'CHMAX',
       'class'],
      dtype='object')
```

```
[28]
#Use LinearRegression for both datasets: model = LinearRegression(), model.fit(), model.score() with right parameters.
#You can use different parameters and check the results. Your task is to predict class.
#What are the differences in the scores of the models? What is the mathematical complexity of the models?


from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split


dataset = pd.read_csv('cpu_vendor.csv')

dataset = pd.concat([pd.get_dummies(dataset['vendor']), dataset.drop('vendor', axis=1)], axis=1) #get dummies remove key error

 #assign x and y
```

```
[34] x=  dataset.iloc[:, :-1].values  # all but class
print (x)
```

```
[[  1   0   0 ... 256  16 128]
 [  0   1   0 ...  32   8  32]
 [  0   1   0 ...  32   8  32]
 ...
 [  0   0   0 ...   0   2  14]
 [  0   0   0 ...  32   0   0]
 [  0   0   0 ...   0   0   0]]
```

```
[39] print ("Dataset.head() \n ", dataset.head())
```

```
Dataset.head()
   adviser  amdahl  apollo  basf  bti  burroughs  c.r.d  cambex  cdc  dec  \
0        1       0       0     0    0          0      0       0    0    0
1        0       1       0     0    0          0      0       0    0    0
2        0       1       0     0    0          0      0       0    0    0
3        0       1       0     0    0          0      0       0    0    0
4        0       1       0     0    0          0      0       0    0    0

   ...  sperry  sratus  wang  MYCT  MMIN   MMAX  CACH  CHMIN  CHMAX  class
0  ...       0       0     0   125   256   6000   256     16    128    199
1  ...       0       0     0    29  8000  32000    32      8     32    253
2  ...       0       0     0    29  8000  32000    32      8     32    253
3  ...       0       0     0    29  8000  32000    32      8     32    253
4  ...       0       0     0    29  8000  16000    32      8     16    132

[5 rows x 37 columns]
```

```
print("\nFirst 10 Input Values : \n", x[0:10, :])
```

```
First 10 Input Values :
 [[   1    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0  125  256 6000  256   16  128]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   29 8000 32000   32    8   32]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   29 8000 32000   32    8   32]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   29 8000 32000   32    8   32]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   29 8000 16000   32    8   16]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   26 8000 32000   64    8   32]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   23 16000 32000   64   16   32]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   23 16000 32000   64   16   32]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   23 16000 64000   64   16   32]
 [   0    1    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0   23 32000 64000  128   32   64]]
```

```
y = dataset['class'].values
print (y)
y1 = y
y1 = y1.reshape(-1, 1)
print("\n\nFirst 10 Output true value : \n", y1[0:10, :])
```

```
[ 199  253  253  253  132  290  381  381  749 1238   23   24   70  117
   15   64   23   29   22  124   35   39   40   45   28   21   28   22
   28   27  102  102   74   74  138  136   23   29   44   30   41   74
   74   74   54   41   18   28   36   38   34   19   72   36   30   56
   42   34   34   34   34   34   19   75  113  157   18   20   28   33
   47   54   20   23   25   52   27   50   18   53   23   30   73   20
   25   28   29   32  175   57  181  181   32   82  171  361  350  220
  113   15   21   35   18   20   20   28   45   18   17   26   28   28
   31   31   42   76   76   26   59   65  101  116   18   20   20   30
   44   44   82   82  128   37   46   46   80   88   88   33   46   29
   53   53   41   86   95  107  117  119  120   48  126  266  270  426
  151  267  603   19   21   26   35   41   47   62   78   80   80  142
  281  190   21   25   67   24   24   64   25   20   29   43   53   19
   22   31   41   47   99   67   81  149  183  275  382   56  182  227
  341  360  919  978   24   24   24   24   37   50   41   47   25]


First 10 Output true value :
 [[ 199]
 [ 253]
 [ 253]
 [ 253]
 [ 132]
 [ 290]
 [ 381]
 [ 381]
 [ 749]
 [1238]]
```

```
[42] xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 0)

     model = LinearRegression()

     model.fit(xtrain, ytrain) #fit

     r_sq = model.score(xtest, ytest)
```

```
y_pred = model.predict(xtest)
print(f"predicted response x:\n{y_pred}")

y_pred1 = y_pred
y_pred1 = y_pred1.reshape(-1,1)

print("\n   RESULT OF LINEAR REGRESSION PREDICTION : ")
print ("\nFirst 10 Predicted value : \n", y_pred1[0:10, :])
```

```
predicted response x:
[ 34.96219718 -63.62905215  85.10288078 329.79562195  99.05069405
 -70.83748694  78.43773879  48.56398268 216.35643742 157.4287584
  35.49906714 539.38780353  82.80865695 246.45641894 472.57975677
 -48.97262973  72.4732777   -7.13119156 -26.07826145 -60.31571147
  -8.09163848 147.49434049 467.78284969  56.74806962 -15.50528646
 110.55364292 -12.88444902   3.87250729 -34.282316    49.41124133
  64.5477403   -2.64524057   1.40177849 141.05971152 144.78549101
  69.08448032  -3.54689037  37.10263258 -63.20761797 110.89945375
 263.63796024  77.0052194 ]

   RESULT OF LINEAR REGRESSION PREDICTION :

First 10 Predicted value :
 [[ 34.96219718]
 [-63.62905215]
 [ 85.10288078]
 [329.79562195]
 [ 99.05069405]
 [-70.83748694]
 [ 78.43773879]
 [ 48.56398268]
 [216.35643742]
 [157.4287584 ]]
```

```
# predict



print(f"coefficient of determination: {r_sq}")

print(f"intercept: {model.intercept_}")

print(f"slope: {model.coef_}")
```

```
coefficient of determination: 0.7791673058680394
intercept: -76.7635817080922
slope: [-1.21924344e+02 -5.64759266e+00  3.24504704e+01 -2.42820193e+01
  5.95723139e+01 -4.40969929e+01  3.80707825e+01  2.63681516e+01
  1.28558635e+01  1.78652928e+01  2.96932061e+01 -1.08355439e+00
  5.01427746e+01 -4.28363744e+01  2.56831882e+00 -4.98145223e+01
  2.10604755e+01 -2.60765921e+00 -5.56711050e+00  2.62345539e+01
 -1.59187495e+02 -2.52654945e+01 -3.21334833e+00  2.22531103e+01
  2.91592274e+01  2.13129601e+01  2.02428200e+01  5.01648142e+01
  3.64988643e+00  2.18614851e+01  7.17060313e-02  1.93664453e-02
  5.50274655e-03  5.51605598e-01 -2.09115212e+00  1.89842018e+00]
```

5. **[1] What are the regression coefficients? Interpret them (if… then…) - provide 3 interpretations.**
   NOTE: A reliable source link would have been helpful here on how to analize this.

The regression coefficients helps in predicting the value of an unknown variable using a known variable by multiplying the variables in a linear regression,they are estimators for the predictor variable and the corresponding response.

https://www.statology.org/how-to-interpret-regression-coefficients/
The intercept term in a regression table tells us the average expected value for the response variable when all of the predictor variables are equal to zero.

Here the intercept is -79 which is evident that something went wrong in the results, however it should imply when the value of everything else, all the other variables is 0,then the average expected value is such.

There is some confusion on the interpretation of the coefficient since there is no given p value, however, as i understand it, for every column increase from 0 there is .0.78 higher average of the class variable.

Part 6:

What is the main difference between ID3 and CART?

The difference between two algorithms is the difference between H(S) and I_G(S)

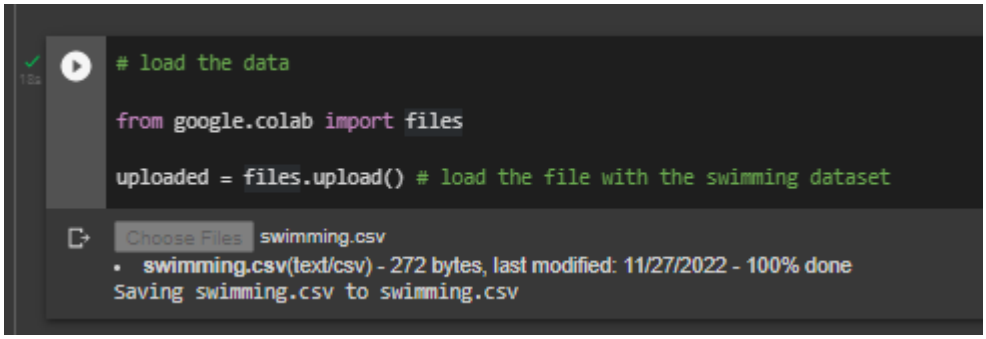
Gini impurity is slightly faster and tends to isolate the most frequent class in its own branch of the tree.
Entropy produces slightly more balanced trees.

What is gini and entropy?

Entropy, is the measure of the amount of uncertainty or randomness in the data. Ranges from 0 (homogeneous) to 1 (perfect randomness).

Gini Impurity tells how mixed the classes are in the two groups created by the split. Ranges from 0 (same lable perfect classification)(good) to 1 (equally distributed among different labels)(bad).

1. **[2] Paste all the code for the swimming example.**



```
[12] from sklearn import tree
     from sklearn.tree import DecisionTreeClassifier
     import pandas as pd
     import numpy as np
     import matplotlib
     import matplotlib.pyplot as plt
     import seaborn as sns
     import missingno
     %matplotlib inline
     from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
     from sklearn.model_selection import train_test_split
```

```
[13] df = pd.read_csv('swimming.csv')
     df.head()
```

	sky	airTemp	humidity	wind	water	forecast	enjoy
0	sunny	warm	normal	strong	warm	same	yes
1	sunny	warm	high	strong	warm	same	yes
2	rainy	cold	high	strong	warm	change	no
3	sunny	warm	high	strong	cool	change	yes
4	cloudy	warm	normal	weak	warm	same	yes

```
[15] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   sky       6 non-null      object
 1   airTemp   6 non-null      object
 2   humidity  6 non-null      object
 3   wind      6 non-null      object
 4   water     6 non-null      object
```

```
df = pd.read_csv('swimming.csv')

df =pd.get_dummies(df)

print(df)
```

```
   sky_cloudy  sky_rainy  sky_sunny  airTemp_cold  airTemp_warm  \
0           0          0          1             0             1
1           0          0          1             0             1
2           0          1          0             1             0
3           0          0          1             0             1
4           1          0          0             0             1
5           1          0          0             1             0

   humidity_high  humidity_normal  wind_strong  wind_weak  water_cool  \
0              0                1            1          0           0
1              1                0            1          0           0
2              1                0            1          0           0
3              1                0            1          0           1
4              0                1            0          1           0
5              1                0            0          1           1

   water_warm  forecast_change  forecast_same  enjoy_no  enjoy_yes
0           1                0              1         0          1
1           1                0              1         0          1
2           1                1              0         1          0
3           0                1              0         0          1
4           1                0              1         0          1
5           0                0              1         1          0
```

```
df.keys()
```

```
Index(['sky_cloudy', 'sky_rainy', 'sky_sunny', 'airTemp_cold', 'airTemp_warm',
       'humidity_high', 'humidity_normal', 'wind_strong', 'wind_weak',
       'water_cool', 'water_warm', 'forecast_change', 'forecast_same',
       'enjoy_no', 'enjoy_yes'],
      dtype='object')
```

```
pd.get_dummies(df.drop('enjoy', axis=1)) # what is going on here?
#print pd without enjoy column
```

	sky_cloudy	sky_rainy	sky_sunny	airTemp_cold	airTemp_warm	humidity_high	humidity_normal	wind_strong	wind_weak	water_cool	water_warm	forecast_change	forecast_same
0	0	0	1	0	1	0	1	1	0	0	1	0	1
1	0	0	1	0	1	1	0	1	0	0	1	0	1
2	0	1	0	1	0	1	0	1	0	0	1	1	0
3	0	0	1	0	1	1	0	1	0	1	0	1	0
4	1	0	0	0	1	0	1	0	1	0	1	0	1
5	1	0	0	1	0	1	0	0	1	1	0	0	1

```
y = df[['enjoy']]
print (y)

y= pd.get_dummies(df[['enjoy']]) # what is going on here?
print (y)
```

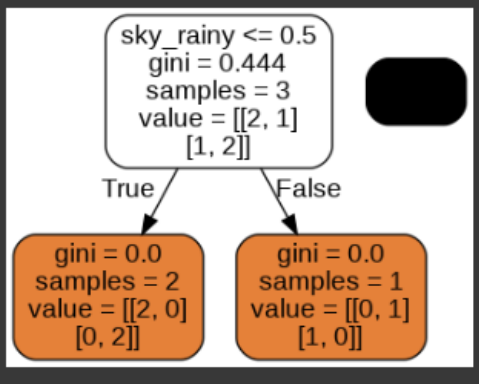
```
  enjoy
0   yes
1   yes
2    no
3   yes
4   yes
5    no
   enjoy_no  enjoy_yes
0         0          1
1         0          1
2         1          0
3         0          1
4         0          1
5         1          0
```

```
[ ]  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.4, random_state = 42)
     #do not understand the implication of test size and random state, works without as well

     clf_gini = DecisionTreeClassifier(criterion='gini')
     clf_entropy = DecisionTreeClassifier(criterion='entropy')

     clf_gini = clf_gini.fit(x_train,y_train)
     clf_entropy  = clf_entropy.fit(x_train,y_train)
```

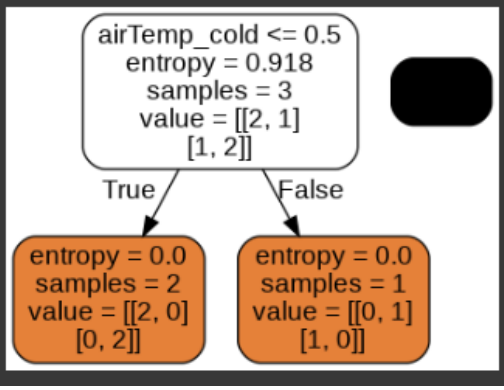
```
[ ]  df = pd.read_csv('swimming.csv') # displays rows and columns of dataset
     pd.get_dummies(df.drop('enjoy', axis=1)) # what is going on here?
```

	sky_cloudy	sky_rainy	sky_sunny	airTemp_cold	airTemp_warm	humidity_high	humidity_normal	wind_strong	wind_weak	water_cool	water_warm	forecast_change	forecast_same
0	0	0	1	0	1	0	1	1	0	0	1	0	1
1	0	0	1	0	1	1	0	1	0	0	1	0	1
2	0	1	0	1	0	1	0	1	0	0	1	1	0
3	0	0	1	0	1	1	0	1	0	1	0	1	0
4	1	0	0	0	1	0	1	0	1	0	1	0	1
5	1	0	0	1	0	1	0	0	1	1	0	0	1

```
# use the following code to visualize the tree # CLF GINI

clf= clf_gini

columns = pd.get_dummies(df.drop('enjoy', axis=1)).columns

dot_data = tree.export_graphviz(clf, out_file=None, rounded=True, filled=True, feature_names=columns)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```




```
# use the following code to visualize the tree # CLF ENTROPHY

clf= clf_entropy

columns = pd.get_dummies(df.drop('enjoy', axis=1)).columns

dot_data = tree.export_graphviz(clf, out_file=None, rounded=True, filled=True, feature_names=columns)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



Unable to function in time trouble shooting

```
# predict

# predict
#clf_gini.predict([X_train,y_train])
#clf_entropy.predict([X_train,y_train])

x_predict = clf_gini.predict(x_test)
y_predict = clf_gini.predict(y_test)

x_predict = clf_entropy.predict(x_test)
y_predict = clf_entropy.predict(y_test)

# probability of each class
clf_gini.predict_proba([x_train,y_train])
clf_entropy.predict_proba([x_train,y_train])

accuracy_score(y_test,y_predict)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:493: FutureWarning: The featu
Feature names unseen at fit time:
- enjoy_no
- enjoy_yes
Feature names seen at fit time, yet now missing:
- airTemp_cold
- airTemp_warm
- forecast_change
- forecast_same
```

2. **[1] Do we need to use pd.get_dummies() for the iris dataset? Why or why not? Before answering, try doing it!**

   Yes we can choose to use the pd.get_dummies() for the iris data set in column class.

```
df = pd.read_csv('iris.csv')
df.head()
```

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepallength  150 non-null    float64
 1   sepalwidth   150 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   class        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
#df = pd.read_csv('swimming.csv')
df = pd.read_csv('iris.csv')
x = pd.get_dummies(df.drop('class', axis=1)) #enjoy
print(x)

#X = df.iloc[:, :-1].values  # all but enjoy (hold that thought)
#print (X)
```

```
     sepallength  sepalwidth  petallength  petalwidth
0            5.1         3.5          1.4         0.2
1            4.9         3.0          1.4         0.2
2            4.7         3.2          1.3         0.2
3            4.6         3.1          1.5         0.2
4            5.0         3.6          1.4         0.2
..           ...         ...          ...         ...
145          6.7         3.0          5.2         2.3
146          6.3         2.5          5.0         1.9
147          6.5         3.0          5.2         2.0
148          6.2         3.4          5.4         2.3
149          5.9         3.0          5.1         1.8

[150 rows x 4 columns]
```

```
df.keys()
```

```
Index(['sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'class'], dtype='object')
```

```
df.describe()
```

	sepallength	sepalwidth	petallength	petalwidth
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
pd.get_dummies(df.drop('class', axis=1)) # what is going on here?
#print pd without enjoy column
```

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
y = df[['class']]
print (y)

y= pd.get_dummies(df[['class']]) # what is going on here?
print (y)
```

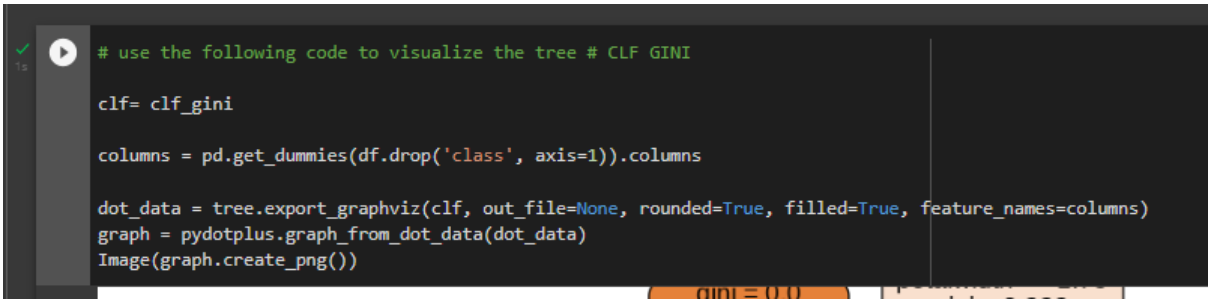
```
            class
0     Iris-setosa
1     Iris-setosa
2     Iris-setosa
3     Iris-setosa
4     Iris-setosa
..            ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

```
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 1 columns]
     class_Iris-setosa  class_Iris-versicolor  class_Iris-virginica
0                    1                      0                     0
1                    1                      0                     0
2                    1                      0                     0
3                    1                      0                     0
4                    1                      0                     0
..                 ...                    ...                   ...
145                  0                      0                     1
146                  0                      0                     1
147                  0                      0                     1
148                  0                      0                     1
149                  0                      0                     1

[150 rows x 3 columns]
```

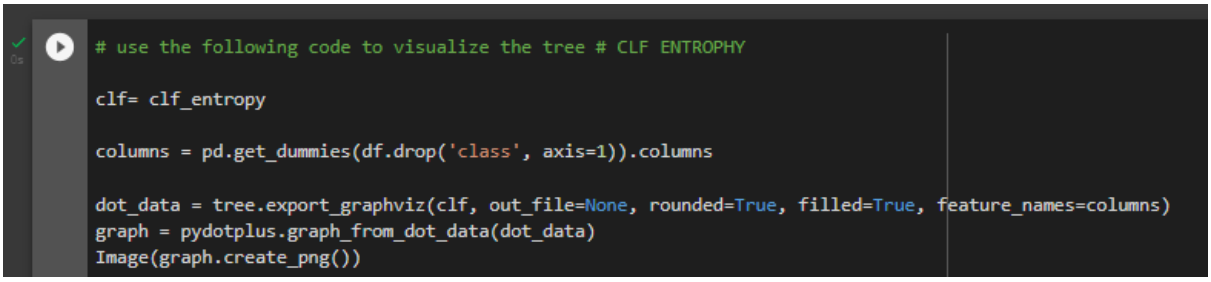
```
df = pd.read_csv('iris.csv') # displays rows and columns of dataset
pd.get_dummies(df.drop('class', axis=1)) # what is going on here?
```

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns



```
# use the following code to visualize the tree # CLF GINI

clf= clf_gini

columns = pd.get_dummies(df.drop('class', axis=1)).columns

dot_data = tree.export_graphviz(clf, out_file=None, rounded=True, filled=True, feature_names=columns)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



```
# use the following code to visualize the tree # CLF ENTROPHY

clf= clf_entropy

columns = pd.get_dummies(df.drop('class', axis=1)).columns

dot_data = tree.export_graphviz(clf, out_file=None, rounded=True, filled=True, feature_names=columns)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

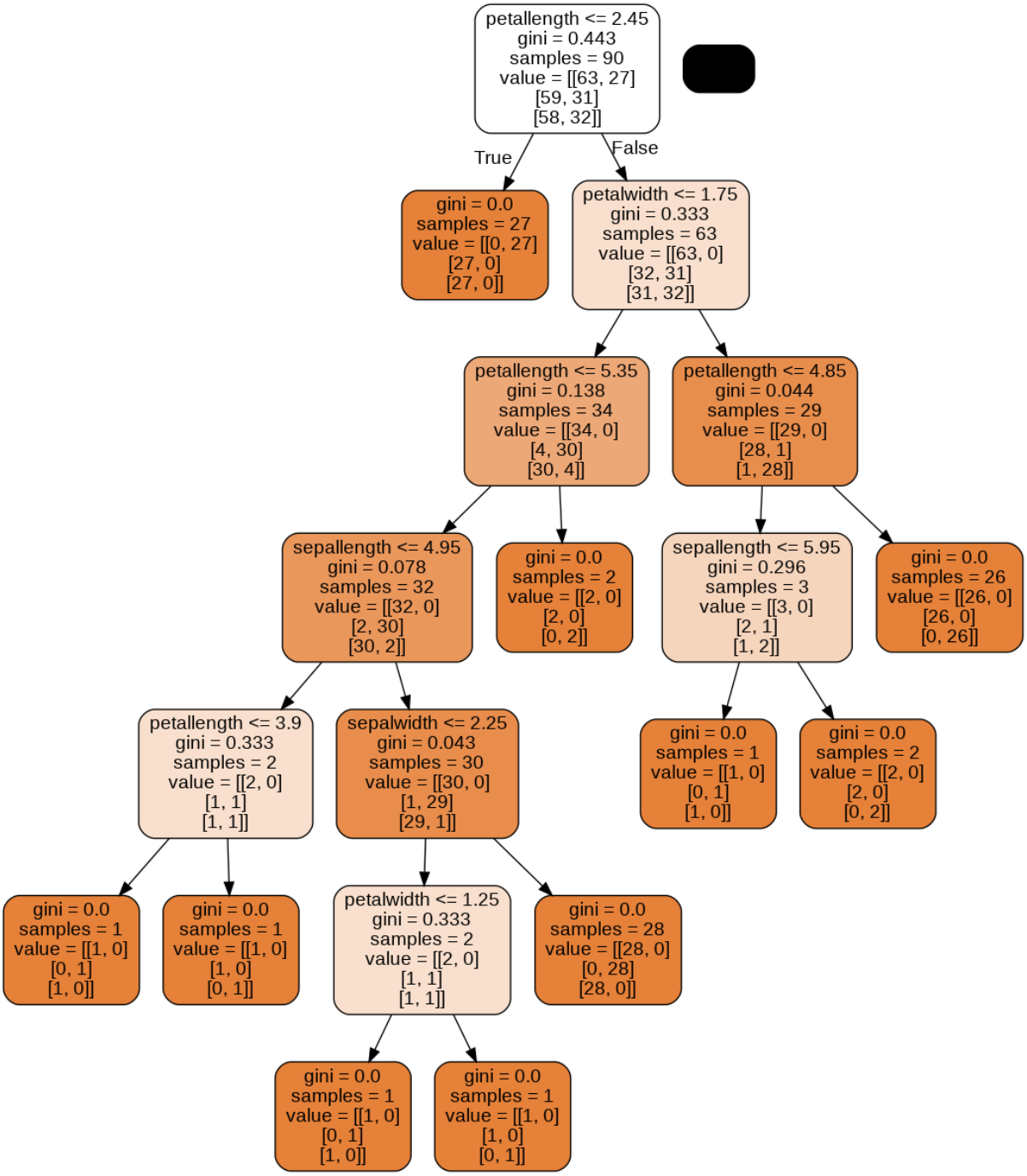
3. **[1] What do the max_depth, min_samples_split, min_samples_leaf parameters do?**

max_depth limits the number of levels deep a decision tree can go. ie. *The tree is not allowed to split any more than a certain number of specified levels and has to end all nodes in a leaf by that level.*

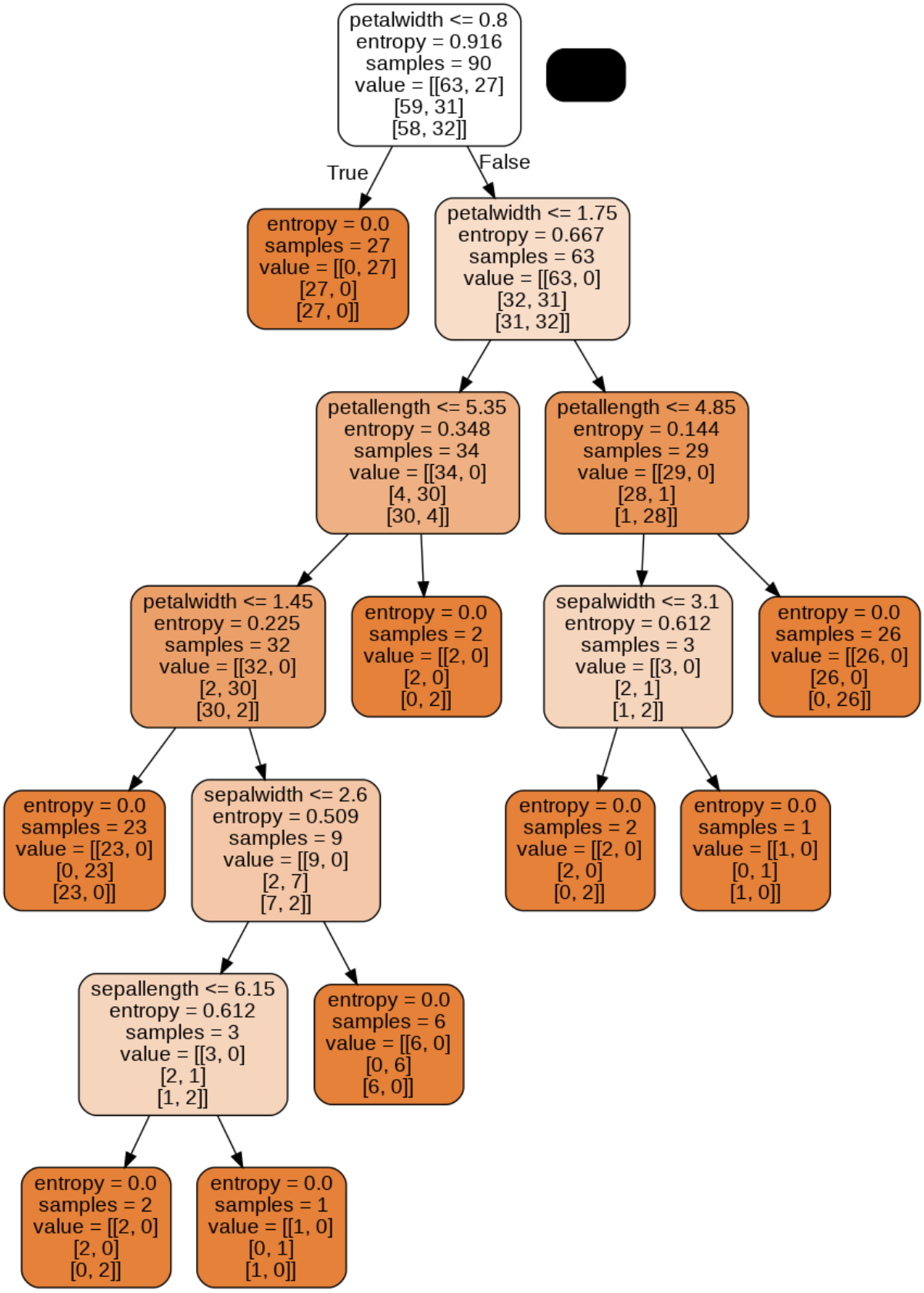
min_samples_split specifies the minimum number of samples required to split an internal node ie. How many times is the algorithm allowed to split the samples, *if the quantity of the given node split is lower than specified samples then the operation will not execute, the node will be a leaf.*

min_samples_leaf parameters specifies the minimum number of samples required to be at a leaf node, it guarantees a minimum number of samples in every leaf, ie.*the program cannot execute if the node split results in a leaf quantity that is lower than the specified min_samples_leaf quantity, the node will be replaced by a single leaf.*

4. **[1] Put graphical representations of trees for the iris dataset (at least one figure per modified parameter – enter values).**

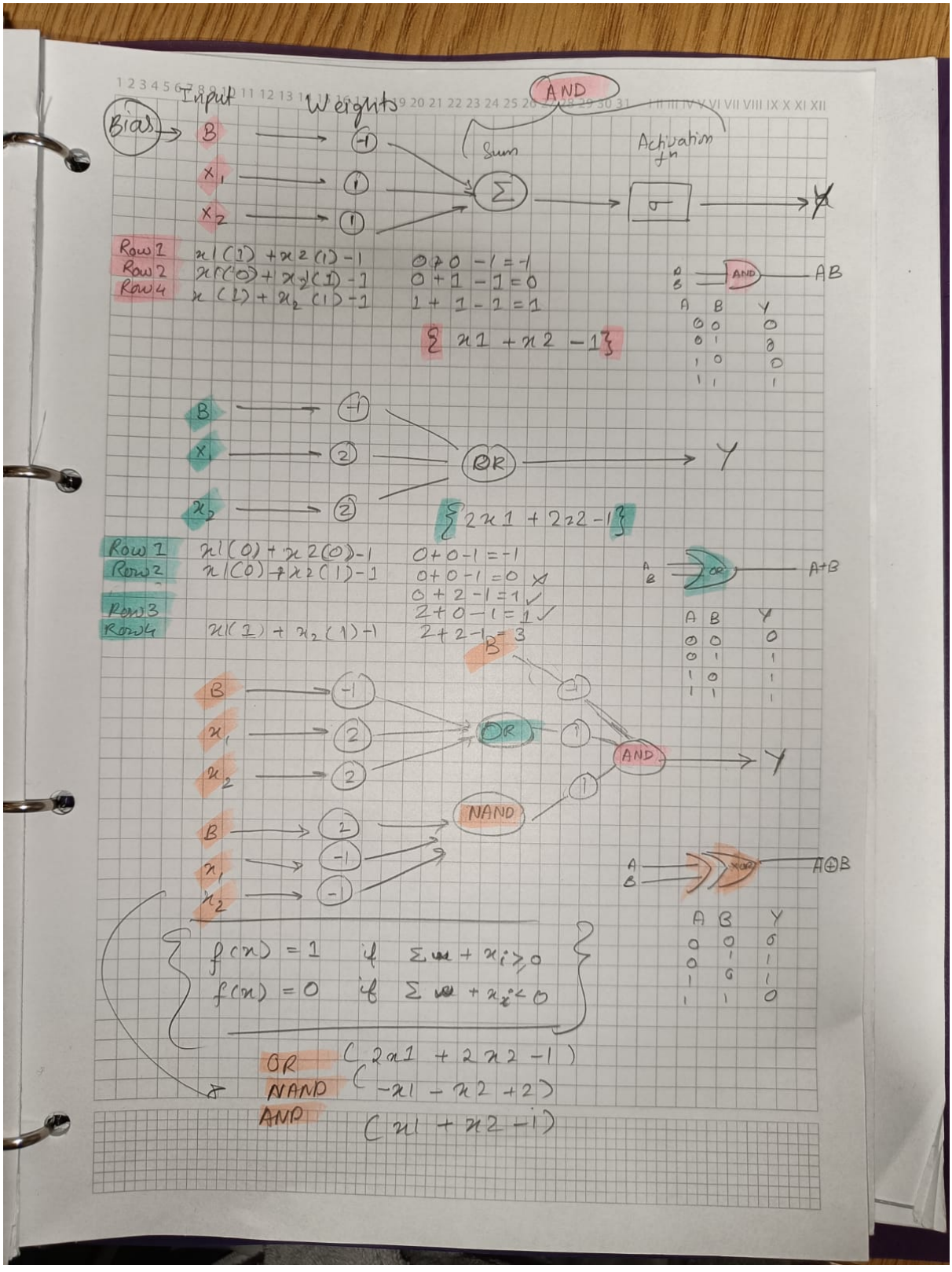


5.

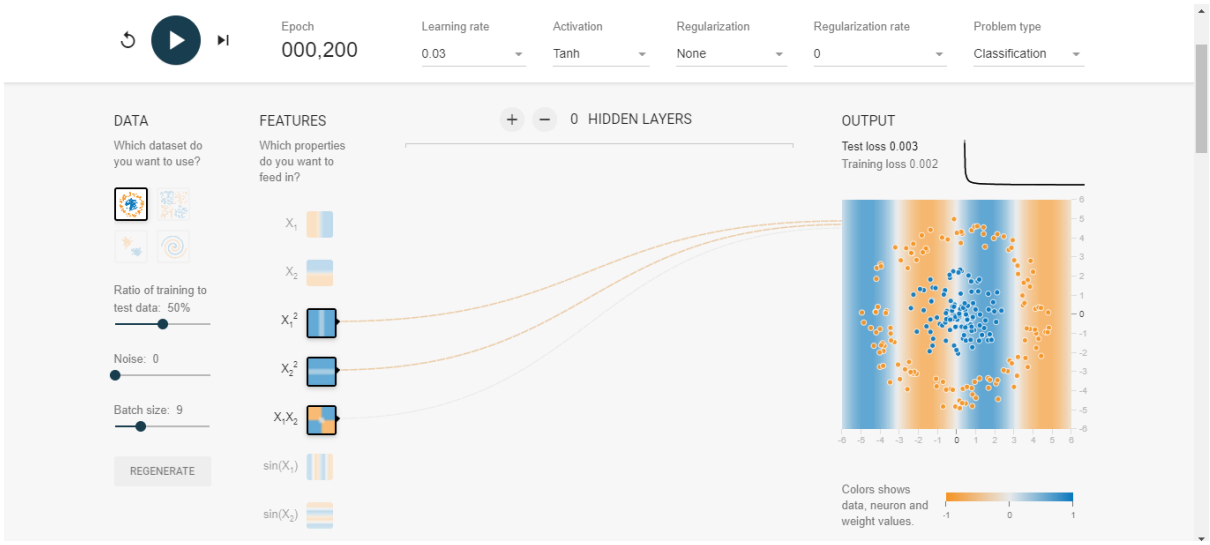


Part 7:

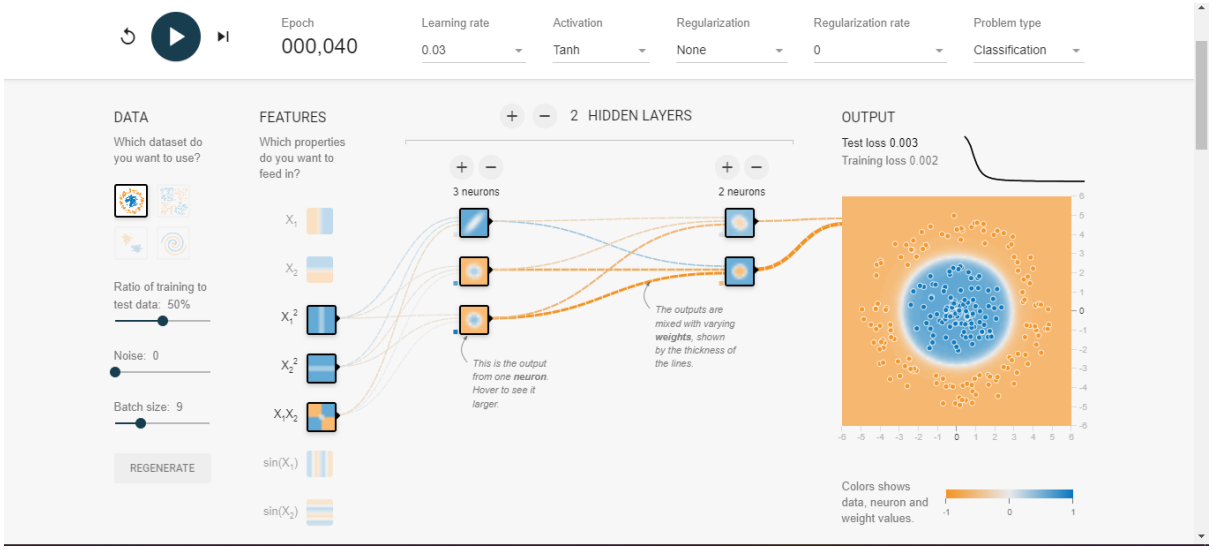
1. **[1] Paste the pictures of the two neurons for AND and OR. Mind plus and minus!**



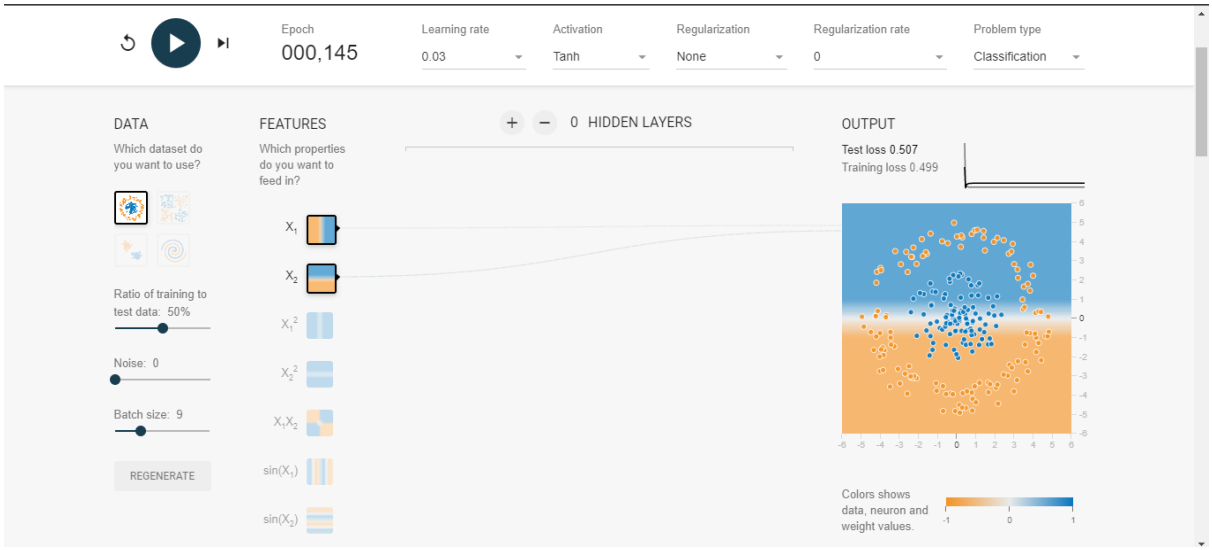
2. **[1] Paste the screenshots of the trained networks for the four datasets from the indicated page (playground).**
   Here in this set the training was not successful: because it was unable to properly group the points as expected by all the other feature inputs



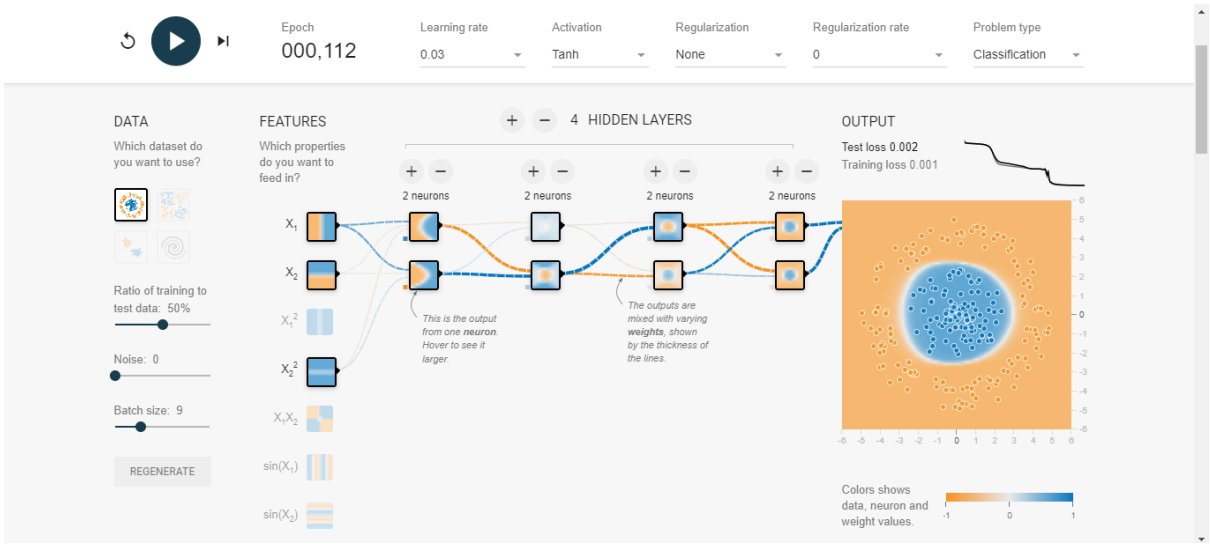
The same input features can have a correct output if two hidden layers are added



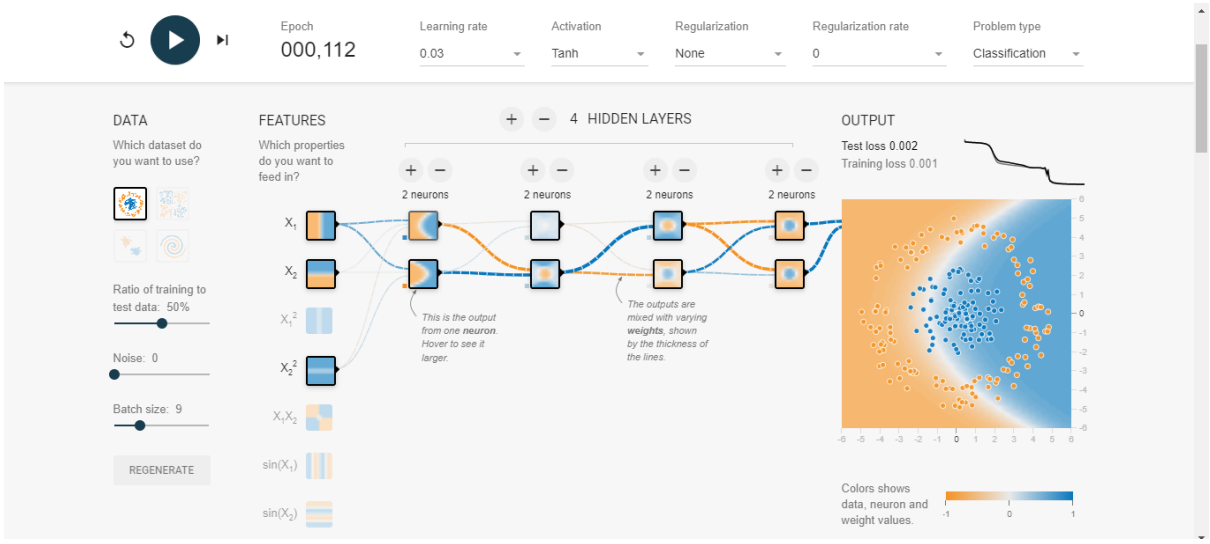
Another simpler example of the result not being as expected



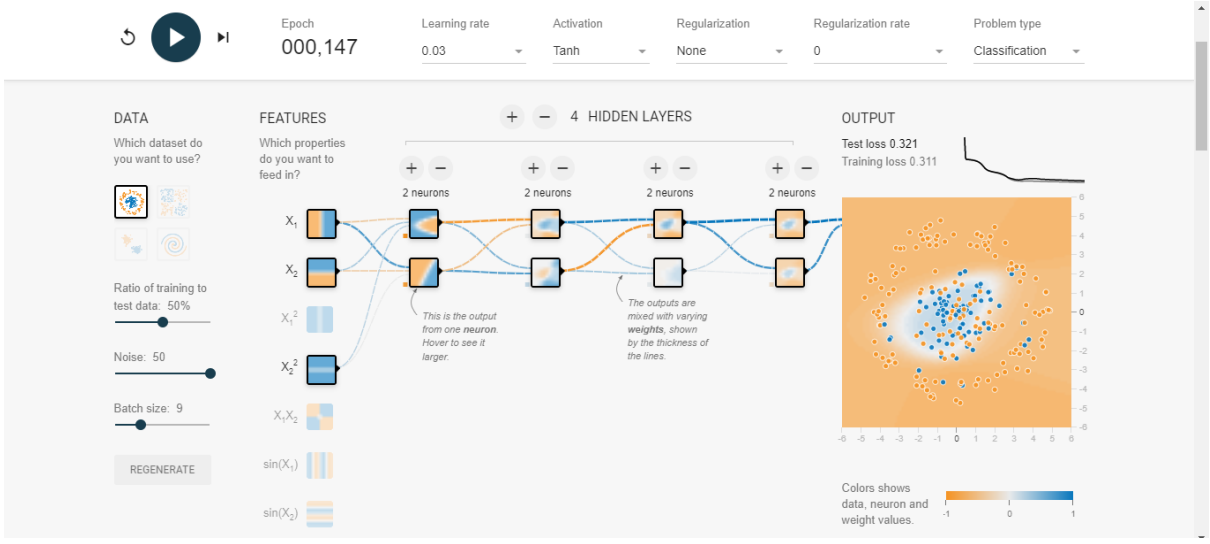
Here with some fun experimentation, adding 4 layers and another input parameter does it job to a satisfactory level.



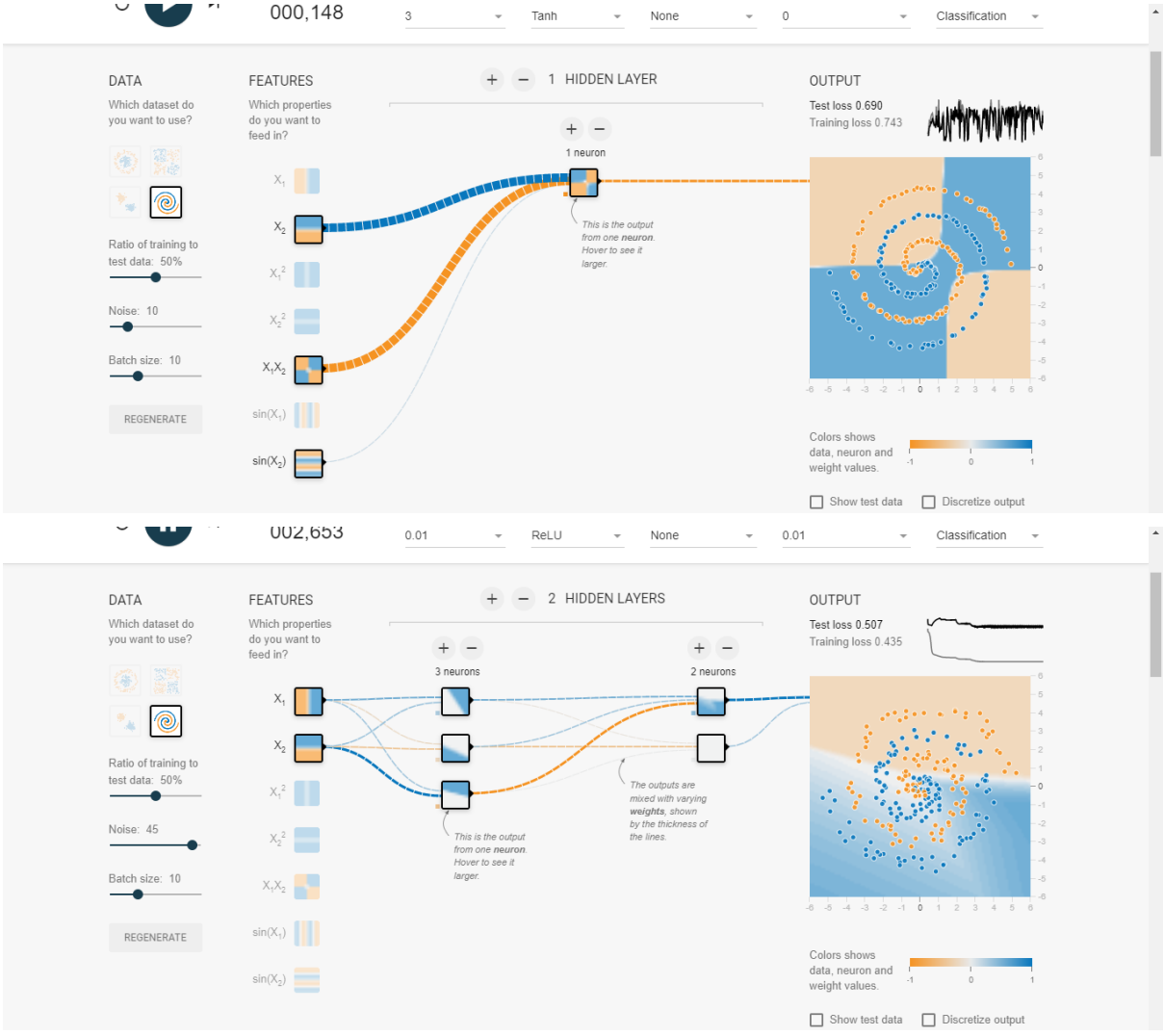
Here the pattern for the first layer first neuron is being checked from the above example



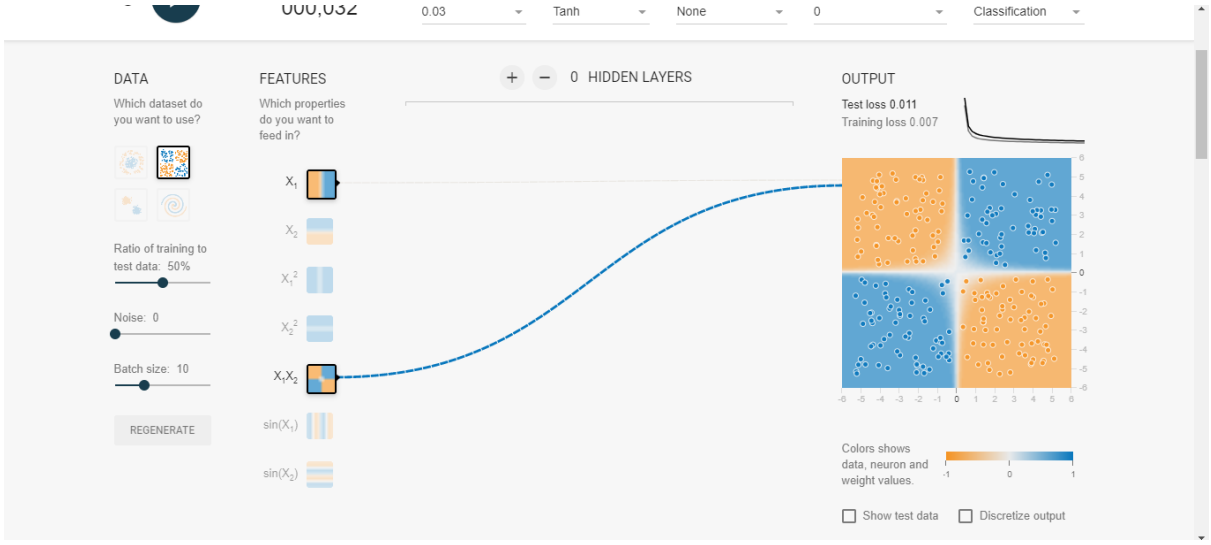
Having added the largest amount of noise possible it becomes evident why we need normalization and other functions to get better grouping



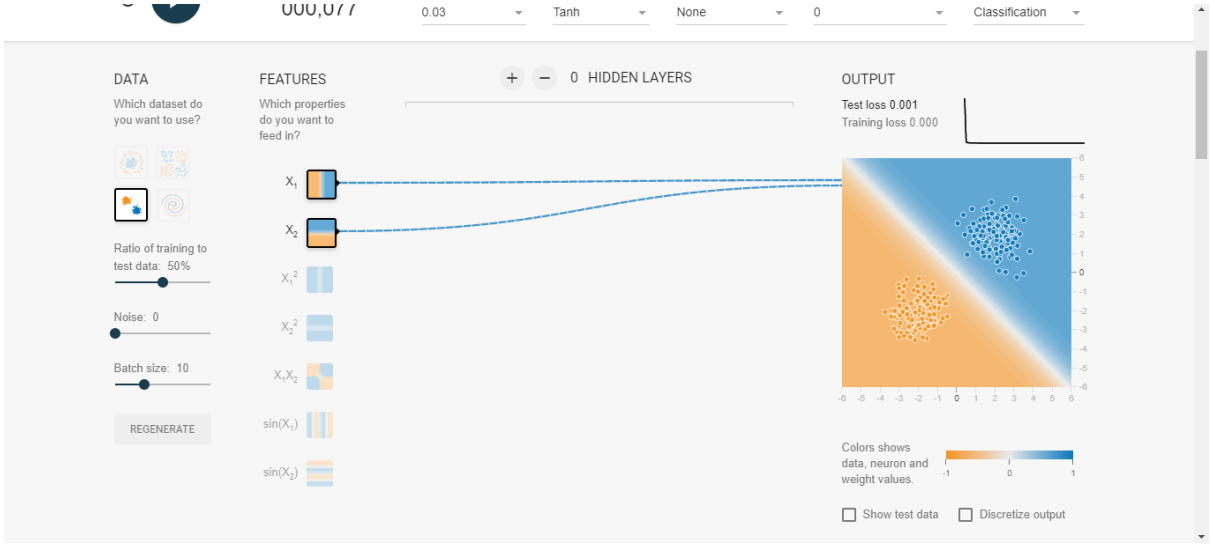
a few attempts were made at training the spiral with as little input and hidden layers as possible however after quite some time and experimentation i was unable to get to a good conclusion

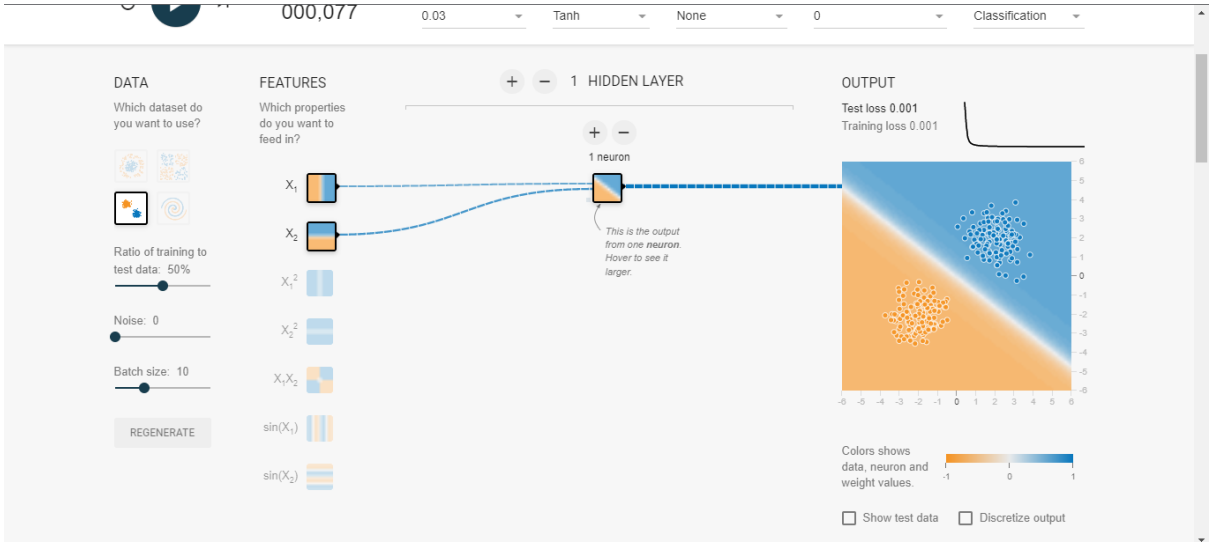


trained model for exclusive or

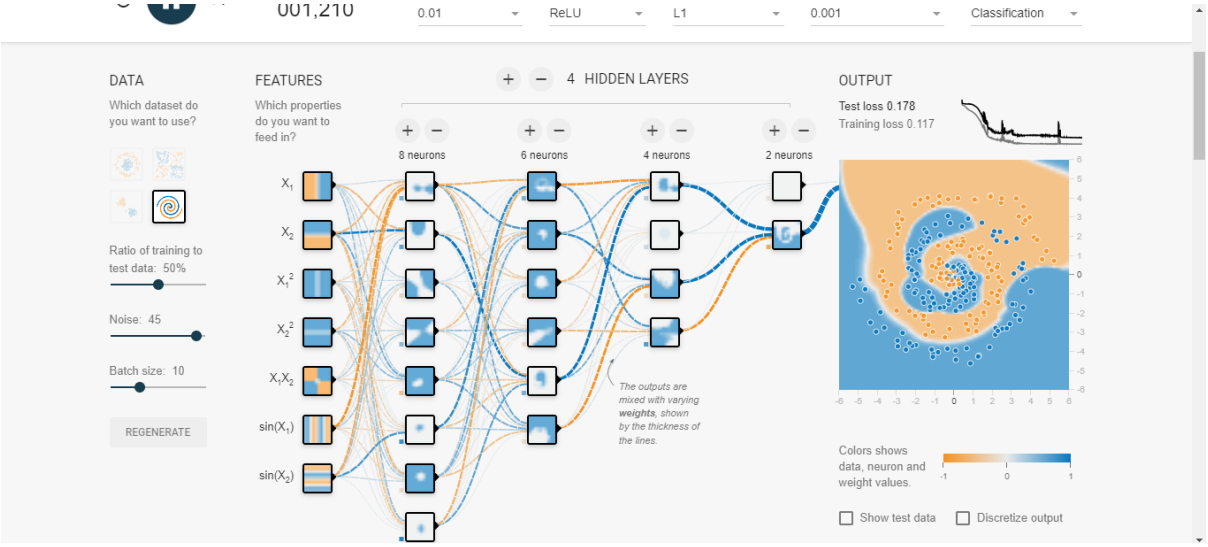


trained model for gaussian





Trained spiral



3. **[1] What is the function train_test_split()? Why are we using it?**
   train_test_split() function has been used to split our dataset into train and test sets.
   X_train, X_test, y_train, y_test = train_test_split(x,y)
   where x is the features and y is the labels, we can also specify parameters such as random_state, test_size and shuffle train_size and stratify.
   It returns split lists.

4. **[1] Which function is responsible for adding layers?**

```
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid' ))
```

   .add()

5. **[1] What does batch_size mean? What does the number of epochs mean? What are these parameters in our example?**
6. **Batch_size** is simply the number of training samples in one forward or backward pass propagating through the network.
7. An **epoch** is one forward pass and one backward pass of all the training examples ie. one has to cycle through all batches to make one epoch in the network.

   NB_EPOCHS = 1000
   BATCH_SIZE = 16
   meaning that the 16 batches were run a thousand times.

8. **[1] What activation functions were used in the exercise? What is the activation function?**

   The **activation function** calculates the weighted sum and adds bias to it, hereby deciding if the neuron should be active or not, this is done so to introduce non-linearity into the output of a neuron.

Some common activation functions are:

- Linear Function
- Sigmoid Function
- Tanh Function
- RELU Function
- Softmax Function

from which we used
Linear Function, Sigmoid Function, Tanh Function, RELU in the nn playground.
in the code we used only relu and sigmoid.

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import missingno
%matplotlib inline
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.model_selection import train_test_split

from IPython.display import Image
import pydotplus

#I know there are redundant import functions in here but everytime i remove something
#the code gets riddled with errors
```

```
[6] # load the file
from google.colab import files

uploaded = files.upload() # load the file with teh swimming dataset
```

Choose Files diabetes.csv

- diabetes.csv(text/csv) - 23098 bytes, last modified: 11/27/2022 - 100% done

Saving diabetes.csv to diabetes.csv

```
[7] df = pd.read_csv('diabetes.csv')
df.head()
```

	pregnancies	glucose	blood_pressure	skin_thickness	insulin	bmi	pedigree_function	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
# check type of columns in df
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   pregnancies        768 non-null    int64  
 1   glucose            768 non-null    int64  
 2   blood_pressure     768 non-null    int64  
 3   skin_thickness     768 non-null    int64  
 4   insulin            768 non-null    int64  
 5   bmi                768 non-null    float64
 6   pedigree_function  768 non-null    float64
 7   age                768 non-null    int64  
 8   class              768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
[13] # count the number of instances, calculate means, variances, etc.
     df.describe()
```

	pregnancies	glucose	blood_pressure	skin_thickness	insulin	bmi	pedigree_function	age	class
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
# show how many classes are there
print(df['class'])
```

```
0      1
1      0
2      1
3      0
4      1
     ..
763    0
764    0
765    0
766    1
767    0
Name: class, Length: 768, dtype: int64
```

```
[14] x = df.drop('class', axis=1).values
     y = df['class'].to_numpy()
```

```
# check type of x and y
print(type(x))
print(type(y))

df.dtypes
```

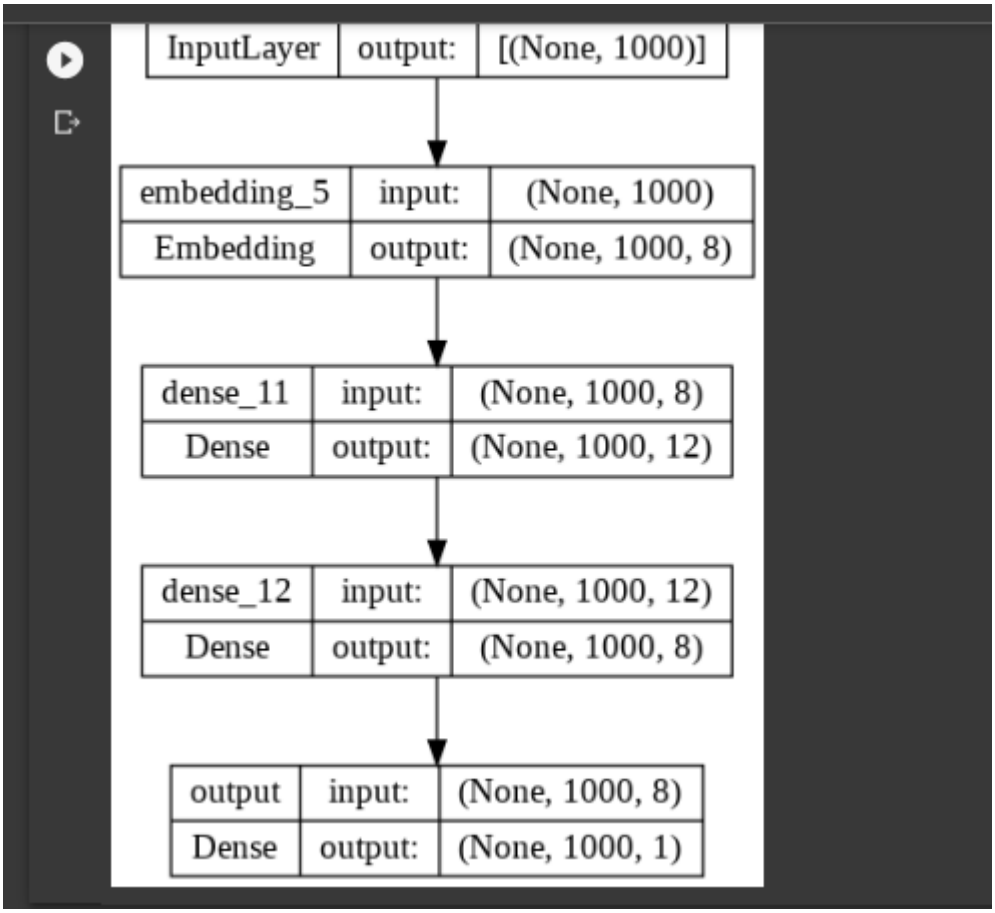
```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
pregnancies              int64
glucose                  int64
blood_pressure           int64
skin_thickness           int64
insulin                  int64
bmi                    float64
pedigree_function      float64
age                      int64
class                    int64
dtype: object
```

```
[20] from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0) # what is going on here?
```

```
### NEURAL NETWORK TRAINING ###

# you do not have to change anything here (for now), just run it

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

NB_EPOCHS = 1000
BATCH_SIZE = 16

model = Sequential()

# adding layers
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid' ))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy']) # metrics

print('Starting training...')

history = model.fit(X_train,
                    y_train,
                    validation_data=(X_test, y_test),
                    epochs=NB_EPOCHS,
                    batch_size=BATCH_SIZE,
                    verbose=1)
```

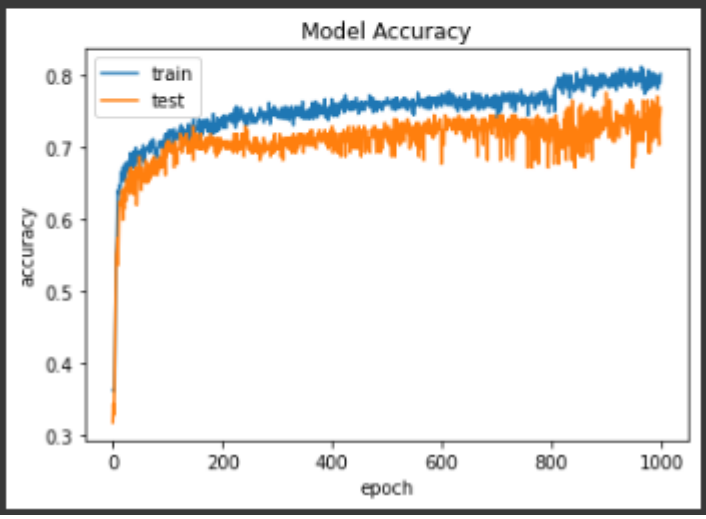
```
Epoch 400/1000
36/36 [==============================] - 0s 3ms/step - loss: 0.4452 - accuracy: 0.7674 - val_loss: 0.7471 - val_accuracy: 0.71
Epoch 401/1000
36/36 [==============================] - 0s 3ms/step - loss: 0.4535 - accuracy: 0.7448 - val_loss: 0.7334 - val_accuracy: 0.72
Epoch 402/1000
36/36 [==============================] - 0s 3ms/step - loss: 0.4416 - accuracy: 0.7604 - val_loss: 0.7346 - val_accuracy: 0.69
Epoch 403/1000
36/36 [==============================] - 0s 3ms/step - loss: 0.4494 - accuracy: 0.7431 - val_loss: 0.7520 - val_accuracy: 0.72
Epoch 404/1000
```

```python
# visualisation

import tensorflow as tf
input = tf.keras.Input(shape=(1000,), dtype='int32', name='class')

x = tf.keras.layers.Embedding(output_dim=8, input_dim=1000, input_length=100)(input)

x = tf.keras.layers.Dense(12,input_dim=1000, activation='relu')(x)

x = tf.keras.layers.Dense(8, activation='relu')(x)

output = tf.keras.layers.Dense(1, activation='sigmoid', name='output')(x)

model = tf.keras.Model(inputs=[input], outputs=[output])

img_file = './model_arch.png'
tf.keras.utils.plot_model(model, to_file=img_file, show_shapes=True, show_layer_names=True)
```



[Part 8:]

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```



1. **[2] Justify by posting a figure and answer the following questions:**
    a. **Are we dealing with *overfitting* or *underfitting*, or is everything just fine?**

This model has been overfitted slighly, but the difference is not massive.

    b. **How many percent of cases were classified correctly?**

Training set reached around 80%, the results diverged around the ~50th epoch and the final difference was ~5%

2. **[1] Is this a good result? Refer to the model's performance if *it guesses* the answer.**

For real life applications it is a decent result, the gap can be further narrowed down but it is acceptable.

3. **[2] Use other metrics (at least 2) and present the results.**

Part 9:

1. **[0] Share with us!**
2. **Sources and other interesting sites during research:**

Note here: these articles are *not* student friendly as they require paywall bypassing or membership (personally i used 12ft.io but not everyone may be aware of such sources)
https://towardsdatascience.com/machine-learning-for-beginners-an-introduction-to-neural-networks-d49f22d238f9
https://towardsdatascience.com/a-brief-introduction-to-supervised-learning-54a3e3932590

https://towardsdatascience.com/simple-and-multiple-linear-regression-in-python-c928425168f9

Main source section 1: class notes
https://www.educba.com/regression-vs-classification/
https://www.ibm.com/cloud/learn/neural-networks
The problem of Overfitting in Regression and how to avoid it? | by Dheeraj Kumar K | DataDrivenInvestor
What is Exploratory Data Analysis? | IBM
https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right
https://www.justintodata.com/machine-learning-model-evaluation-metrics/

Overall:
https://machinelearningknowledge.ai/decision-tree-classifier-in-python-sklearn-with-example/

https://www.statology.org/pandas-get-dummies/
https://gist.github.com/jiahao87/c97214065f996b76ab8fe4ca1964b2b5
https://www.analyticsvidhya.com/blog/2021/05/exploratory-data-analysis-eda-a-step-by-step-guide/

Somewhere along the way the sources were far too many to list so these are just the initial ones. Acknowledging the rightful owners, writers, researchers and codeers of all sources, websites and code repositories.

Part 10:

1. **[0] Share your impressions from the classes. Was there anything too easy/hard/slow/fast/boring/etc.?**
   First i would like to appreciate the understanding for the deadline extensions, the past few weeks have been stressful and knowing that i still had a bit more time allowed for a better understanding of the material.

   Colab, Jupiter notebooks, cell and the **entire work interface was very optimized**, i am quite happy with it, having the inbuilt code compiler and result is very convenient. being able to easily work remotely on the assignment was a big plus.

   The learning curve for me overall was unexpectedly steep, i am used to being able to grasp concepts quickly however here i had to do a lot of research to understand what we were doing, having more verbose written coding instructions may be helpful in that regard.

   The nn playground by far was the most fun and best experience, if there can be more exercises based on that i would like to reproduce more fitted models.

2. **[0] What would you improve?**

Video series introduction if possible, it took me quite a while to get the hang of the prgrams and what to do, while the research alone was enjoyable it was also stressful due to the timelimits. (Video series covering the basics of the commands in python and another for the overall what we are attempting to do, i tried to glace at youtube for something but ended up clicking off due to not having time to go into too much detail, so a short series/presentation would be appreciated)

Additionally: so that other students do not repeat my mistakes, it should be stressed on how much time these expersizes are expected to take, despite asking for a lot of extra time from my end which i do feel guilty about, I know that somewhere my report will be lacking.

What was missing?

A minor mistake: few of the words in nn were in polish in the notebook.
As mentioned previously, a few of the articles form **towardsdatascience** are not student friendly as i had to use a paywall by pass.
A few diagrams of the expected output would have been really helpful.

Is there anything that you believe should be covered more thoroughly?

I believe that there should have been perhaps a recorded tutorial on at least the starting exercises, available also after class as in my case i had to leave a few minutes earlier and its also easy to forget what was explained if the problems have intricate solutions and many sources of information.

Did you get interested in the topic?

Yes, i liked the neural network simulation a lot for example.

Would you like to know more?

For sure, it was an interesting experience.

Was it worth it?

The report was exceedingly long at the end but a lot of research was done and i do believe i learned something new.

Thank you for the presentations and checking our work!

Apoorva Singh
409428