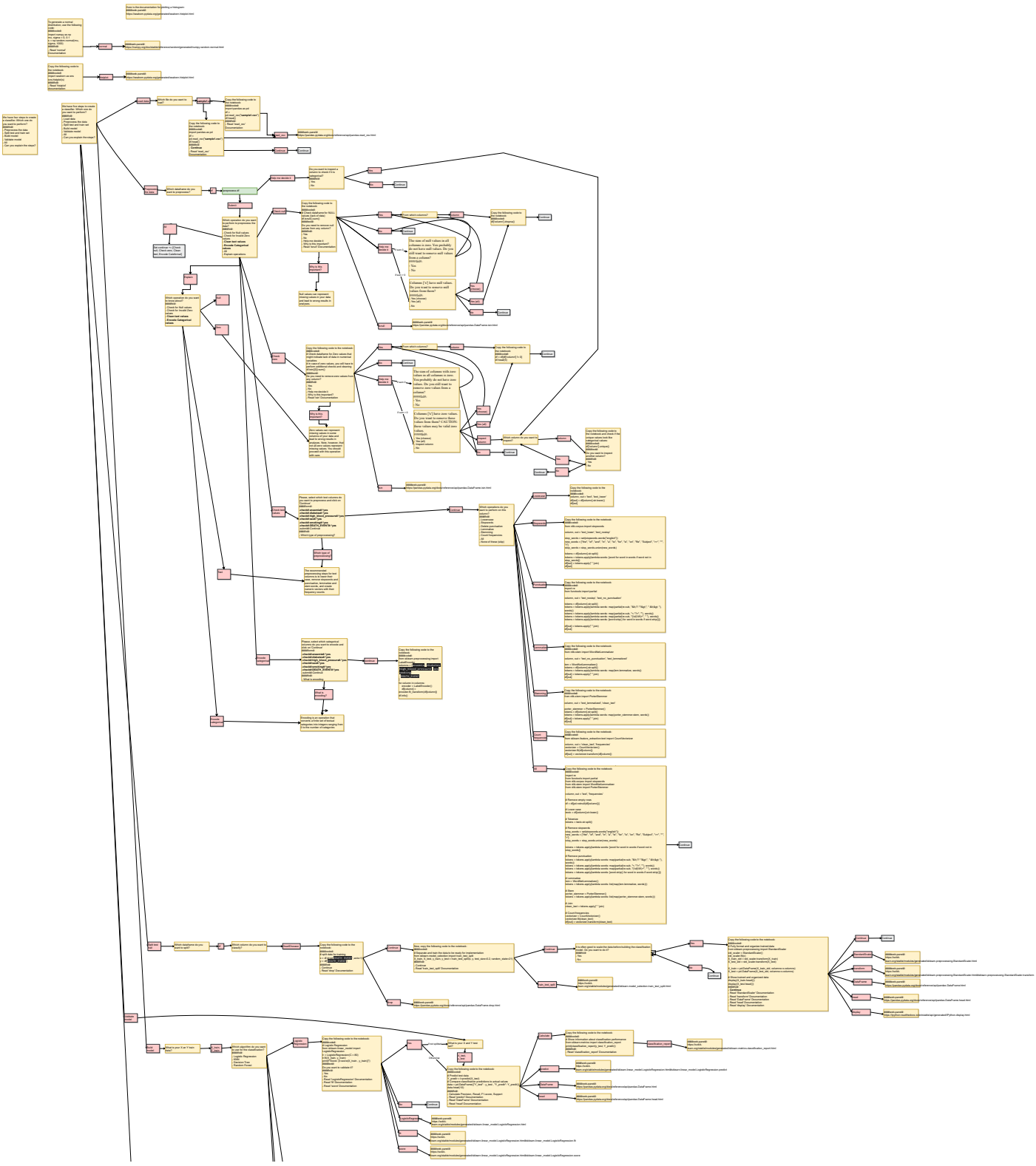
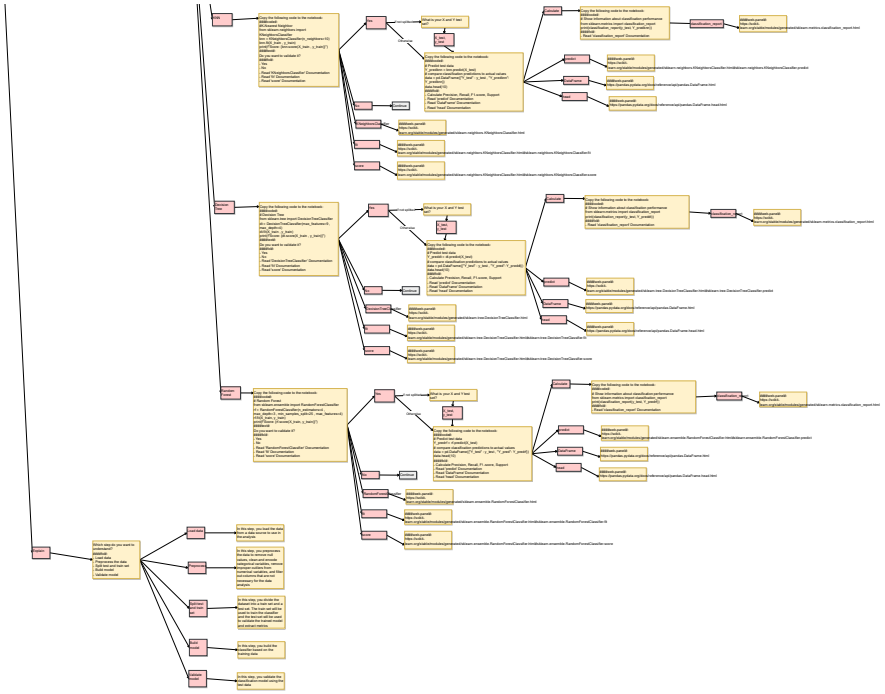
Newton WoZ Script

1 Decision Tree

During the experiments, we used the script mapped as the following decision graph:





2 Text version

We represent it textually below:

>>> User query: “classification”

Go to 1

1. Newton message:

We have five steps to create a classifier. Which one do you want to perform?
####fol#:
– Load data
– Preprocess the data
– Split test and train set
– Build model
– Validate model
– All
– Can you explain the steps?

1.A. Newton-message: [Context: after load data]

We have four steps to create a classifier. Which one do you want to perform?
####fol#:
– Preprocess the data
– Split test and train set
– Build model
– Validate model
– All
– Can you explain the steps?

>>> User enactable suggestion: Load data

Go to 2

>>> User enactable suggestion: Preprocess the data

Go to 5 [Context: no dataframe]

Go to 6 [Context: single dataframe in notebook]

>>> User enactable suggestion: Split test and train set

Go to 41 [Context: no dataframe]

Go to 42 [Context: single dataframe in notebook]

>>> User enactable suggestion: Build model

Go to 54 [Context: before split]

Go to 55 [Context: after split]

>>> User enactable suggestion: Validate model

Go to 58

>>> User enactable suggestion: All

Set [Continue] = [2, 6, 42, 55] (Note: load data 2 may not be available)

>>> User enactable suggestion: Can you explain the steps

Go to 100

2. Newton message:

```
Which file do you want to load?
```

>>> User reply: [filename]

Set [filename]

Go to 3

3. Newton message:

```
Copy the following code to the notebook:
####code#:
import pandas as pd
df = pd.read_csv("<filename>")
df.head()
####fol#:
- Read 'read_csv' Documentation
```

3.A. Newton-message: [Context: has continue]

```
Copy the following code to the notebook:
####code#:
import pandas as pd
df = pd.read_csv("<filename>")
df.head()
####fol#:
- Continue
- Read 'read_csv' Documentation
```

>>> User enactable suggestion: Read 'read_csv' Documentation

Go to 4

>>> User enactable suggestion: Continue

Continue

4. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html
```

5. Newton message:

```
Which dataframe do you want to preprocess?
```

>>> User reply: [dataframe=df]

Set [dataframe]

Go to 3

6. Newton message:

```
Message to Kernel:
!preprocess <dataframe>
```

6.A. Newton-message: [Sample 1]

```
Great! Before we proceed, please set the types of the columns of your dataframe and
    ↪ click on 'Continue'
####form#:
-selection#:age#:Categorical#:*Numeric#:Other
-selection#:anaemia#:Categorical#:Text#:Other
-selection#:creatinine_phosphokinase#:Categorical#:Numeric#:Other
-selection#:diabetes#:Categorical#:Text#:Other
```

```
-selection#:ejection_fraction#:Categorical#:Numeric#:Other
-selection#:high_blood_pressure#:Categorical#:Text#:Other
-selection#:platelets#:Categorical#:*Numeric#:Other
-selection#:serum_creatinine#:Categorical#:*Numeric#:Other
-selection#:serum_sodium#:Categorical#:Numeric#:Other
-selection#:sex#:Categorical#:Text#:Other
-selection#:smoking#:Categorical#:Text#:Other
-selection#:time#:Categorical#:Numeric#:Other
-selection#:DEATH_EVENT#:Categorical#:Text#:Other
-submit#:Continue
####text#:
Or choose an option below
####ol#:
- Help me decide it
- Skip preprocessing
```

6.B. Newton-message: [Sample 2]

```
Great! Before we proceed, please set the types of the columns of your dataframe and
    ↪ click on 'Continue'
####form#:
-selection#:Age#:Categorical#:Numeric#:Other
-selection#:Sex#:Categorical#:Text#:Other
-selection#:ChestPainType#:Categorical#:Text#:Other
-selection#:RestingBP#:Categorical#:Numeric#:Other
-selection#:Cholesterol#:Categorical#:Numeric#:Other
-selection#:FastingBS#:Categorical#:Numeric#:Other
-selection#:RestingECG#:Categorical#:Text#:Other
-selection#:MaxHR#:Categorical#:Numeric#:Other
-selection#:ExerciseAngina#:Categorical#:Text#:Other
-selection#:Oldpeak#:Categorical#:*Numeric#:Other
-selection#:ST_Slope#:Categorical#:Text#:Other
-selection#:HeartDisease#:Categorical#:Numeric#:Other
-submit#:Continue
####text#:
Or choose an option below
####ol#:
- Help me decide it
- Skip preprocessing
```

>>> User enactable suggestion: Help me decide it

Go to 7

>>> User enactable suggestion: Skip preprocessing

Continue to the next step if available. Otherwise, do not reply.

>>> User enactable suggestion: Submit

Go to 10

7. Newton message:

```
Do you want to inspect a column to check if it is categorical?
####fol#:
- Yes
- No
```

>>> User enactable suggestion: Yes

Go to 8

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

8. Newton message:

```
Which column do you want to inspect?
```

>>> User reply: [column]

Set [column]

Go to 9

9. Newton message:

```
Copy the following code to the notebook and check if the unique values look like
    ↪ categorical values:
####code#:
df['[column]'].unique()
####text#:
Do you want to inspect another column?
####fol#:
- Yes
- No
```

>>> User enactable suggestion: Yes

Go to 8

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

10. Newton message:

```
Which operation do you want to perform to preprocess the data?
####fol#:
- Check for Null values
- Check for Invalid Zero values
- Clean text values
- Encode Categorical values
- All
- Explain operations
```

10.A. Newton-message: [Context: does not have text values]

```
Which operation do you want to perform to preprocess the data?
####fol#:
- Check for Null values
- Check for Invalid Zero values
- Encode Categorical values
- All
- Explain operations
```

10.B. Newton-message: [Context: does not have categorical values]

```
Which operation do you want to perform to preprocess the data?
####fol#:
- Check for Null values
- Check for Invalid Zero values
- Clean text values
- All
- Explain operations
```

10.C. Newton-message: [Context: does not have text nor categorical values]

```
Which operation do you want to perform to preprocess the data?
####fol#:
- Check for Null values
- Check for Invalid Zero values
- All
- Explain operations
```

>>> User enactable suggestion: Check for Null values

Go to 11

>>> User enactable suggestion: Check for Invalid Zero values

Go to 19

>>> User enactable suggestion: Clean text values

Go to 27

>>> User enactable suggestion: Encode Categorical values

Go to 37

>>> User enactable suggestion: All

Set [Continue] = [11, 19, 27, 37] (Note: clean text 27 and encode categorical 37 may not be available)

>>> User enactable suggestion: Explain operations

Go to 40

11. Newton message:

```
Copy the following code to the notebook:
####code#:
# Check dataframe for NULL values (lack of data)
df.isnull().sum()
####text#:
Do you need to remove null values from any column?
####fol#:
- Yes
- No
- Help me decide it
- Why is this important?
- Read 'isnull' Documentation
```

>>> User enactable suggestion: Yes

Go to 12

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

>>> User enactable suggestion: Help me decide it

Go to 14 [if sum == 0]

Go to 15 [if sum ¿ 0]

>>> User enactable suggestion: Why is this important?

Go to 17

>>> User enactable suggestion: Read 'isnull' Documentation

Go to 18

12. Newton message:

```
From which columns?
```

>>> User reply: [column]

Set [column]

Go to 13

13. Newton message: [Continue to next step, if available]

```
Copy the following code to the notebook:
####code#:
df = df['[column]'].dropna()
```

14. Newton message:

```
The sum of null values in all columns is zero. You probably do not have null values.
    ↪ Do you still want to remove null values from a column?
####fol#:
- Yes
- No
```

>>> User enactable suggestion: Yes

Go to 12

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

15. Newton message: Set [x] as the columns with sum ¿ 0

```
Columns ['[x]'] have null values. Do you want to remove null values from them?
####fol#:
- Yes (choose)
- Yes (all)
- No
```

>>> User enactable suggestion: Yes (choose)

Go to 12

>>> User enactable suggestion: Yes (all)

Go to 16

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

16. Newton message: [Continue to next step, if available]

```
Copy the following code to the notebook:
####code#:
df = df['[x1]'].dropna()
df = df['[x2]'].dropna()
...
```

17. Newton message:

```
Null values can represent missing values in your data and lead to wrong analysis
    ↪ results.
```

18. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isin.html
```

19. Newton message:

```
Copy the following code to the notebook:
####code#:
# Check dataframe for Zero values that might indicate lack of data in numerical
    ↪ variables
# In case of zero values, you will have to perform additional checks and cleaning
df.isin([0]).sum()
####text#:
Do you need to remove zero values from any column?
####fol#:
- Yes
- No
- Help me decide it
- Why is this important?
- Read 'isin' Documentation
```

>>> User enactable suggestion: Yes

Go to 20

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

>>> User enactable suggestion: Help me decide it

Go to 22 [if sum == 0]

Go to 23 [if sum ¿ 0]

>>> User enactable suggestion: Why is this important?

Go to 25

>>> User enactable suggestion: Read 'isin' Documentation

Go to 26

20. Newton message:

```
From which columns?
```

>>> User reply: [column]

Set [column]

Go to 21

21. Newton message: [Continue to next step, if available]

```
Copy the following code to the notebook:
####code#:
df = df[df['[column]'] != 0]
df.head(5)
```

22. Newton message:

```
The sum of columns with zero values in all columns is zero. You probably do not have
    ↪ zero values. Do you still want to remove zero values from a column?
####fol#:
- Yes
- No
```

>>> User enactable suggestion: Yes

Go to 20

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

23. Newton message: Set [x] as the columns with sum ¿ 0

```
Columns ['[x]'] have zero values. Do you want to remove these values from them?
    ↪ CAUTION: these values may be valid zero values.
####fol#:
- Yes (choose)
- Yes (all)
- Inspect column
- No
```

>>> User enactable suggestion: Yes (choose)

Go to 20

>>> User enactable suggestion: Yes (all)

Go to 24

>>> User enactable suggestion: Inspect column

Go to 8

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

24. Newton message: [Continue to next step, if available]

```
Copy the following code to the notebook:
####code#:
df = df[df['[x1]'] != 0]
df = df[df['[x2]'] != 0]
...
```

25. Newton message:

```
Zero values can represent missing values in some columns of your data and lead to
    ↪ wrong results in analyses. Note, however, that not all zero values represent
    ↪ missing values. You should proceed with this operation with care.
```

26. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isin.html
```

27. Newton message: [The columns here depends on what the user selected in the previous form as text columns]

```
Please, select which text columns do you want to preprocess and click on 'Continue'
####form#:
-check#:[Text column]#:*yes
-submit#:Continue
####fol#:
- Which type of preprocessing?
```

27.A. Newton-message: Sample 1

```
Please, select which text columns do you want to preprocess and click on 'Continue'
####form#:
-check#:anaemia#:*yes
-check#:diabetes#:*yes
-check#:high_blood_pressure#:*yes
-check#:sex#:*yes
-check#:smoking#:*yes
```

−check#:DEATH_EVENT#:∗yes
−submit#:Continue
####fol#:
− Which type of preprocessing?

27.B. Newton-message: Sample 2

Please, select which text columns do you want to preprocess and click on 'Continue'
####form#:
−check#:Sex#:∗yes
−check#:ChestPainType#:∗yes
−check#:RestingECG#:∗yes
−check#:ExerciseAngina#:∗yes
−check#:ST_Slope#:∗yes
−submit#:Continue
####fol#:
− Which type of preprocessing?

>>> User enactable suggestion: Which type of preprocessing?

Go to 28

>>> User enactable suggestion: Submit

Set [columns]

Go to 29

28. Newton message:

The recommended preprocessing steps for text columns is to lower their case, remove stopwords and punctuation, lemmatize and stem words, and create numeric vectors with their frequency counts

29. Newton message:

Which operations do you want to perform on this column?
####fol#:
− Lowercase
− Stopwords
− Delete punctuation
− Lemmatize
− Stemming
− Count frequencies
− All
− None of these (skip)

>>> User enactable suggestion: Lowercase

Go to 30

>>> User enactable suggestion: Stopwords

Go to 31

>>> User enactable suggestion: Delete punctuation

Go to 32

>>> User enactable suggestion: Lemmatize

Go to 33

>>> User enactable suggestion: Stemming

Go to 34

>>> User enactable suggestion: Count frequencies

Go to 35

>>> User enactable suggestion: All

Go to 36

>>> User enactable suggestion: None of these (skip)

Continue to the next step if available. Otherwise, do not reply.

30. Newton message: [After: Set columns = [columns]_lower]

Copy the following code to the notebook:
####code#:

```
column, out = '[columns]', '[columns]_lower'
df[out] = df[column].str.lower()
df[out]
```

31. Newton message: [After: Set columns = [columns]_nostop]

```
Copy the following code to the notebook:
####code#:
from nltk.corpus import stopwords

column, out = '[columns]', '[columns]_nostop'

stop_words = set(stopwords.words("english"))
new_words = ["the", "of", "and", "in", "a", "to", "for", "is", "on", "Re", "Subject
    ↪ ", ">>", "*", "r"]
stop_words = stop_words.union(new_words)

tokens = df[column].str.split()
tokens = tokens.apply(lambda words: [word for word in words if word not in
    ↪ stop_words])
df[out] = tokens.apply(" ".join)
df[out]
```

32. Newton message: [After: Set columns = [columns]_no_punctuation]

```
Copy the following code to the notebook:
####code#:
import re
from functools import partial

column, out = '[columns]', '[columns]_no_punctuation'

tokens = df[column].str.split()
tokens = tokens.apply(lambda words: map(partial(re.sub, "&lt;/?.*?&gt;", " &lt;&gt;
    ↪ "), words))
tokens = tokens.apply(lambda words: map(partial(re.sub, "<.*?>", ""), words))
tokens = tokens.apply(lambda words: map(partial(re.sub, "(\\d|\\W)+", " "), words))
tokens = tokens.apply(lambda words: [word.strip() for word in words if word.strip()
    ↪ ])

df[out] = tokens.apply(" ".join)
df[out]
```

33. Newton message: [After: Set columns = [columns]_lemmatized]

```
Copy the following code to the notebook:
####code#:
from nltk.stem import WordNetLemmatizer

column, out = '[columns]', '[columns]_lemmatized'

lem = WordNetLemmatizer()
tokens = df[column].str.split()
tokens = tokens.apply(lambda words: map(lem.lemmatize, words))
df[out] = tokens.apply(" ".join)
df[out]
```

34. Newton message: [After: Set columns = [columns]_clean]

```
Copy the following code to the notebook:
####code#:
from nltk.stem import PorterStemmer

column, out = '[columns]', '[columns]_clean'

porter_stemmer = PorterStemmer()
tokens = df[column].str.split()
tokens = tokens.apply(lambda words: map(porter_stemmer.stem, words))
```

```
df[out] = tokens.apply(" ".join)
df[out]
```

35. Newton message: [After: Set columns = [columns]_frequencies]

```
Copy the following code to the notebook:
####code#:
from sklearn.feature_extraction.text import CountVectorizer

column, out = '[columns]', '[columns]_frequencies'
vectorizer = CountVectorizer()
vectorizer.fit(df[column])
df[out] = vectorizer.transform(df[column])
```

36. Newton message: [After: Set columns = [columns]_frequencies]

```
Copy the following code to the notebook:
####code#:
import re
from functools import partial
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer

column, out = '[columns]', '[columns]_frequencies'

# Remove empty rows
df = df[pd.notnull(df[column])]

# Lower case
textc = df[column].str.lower()

# Tokenize
tokens = textc.str.split()

# Remove stopwords
stop_words = set(stopwords.words("english"))
new_words = ["the", "of", "and", "in", "a", "to", "for", "is", "on", "Re", "Subject
    ↪ ", ">>", "*", "r"]
stop_words = stop_words.union(new_words)

tokens = tokens.apply(lambda words: [word for word in words if word not in
    ↪ stop_words])

# Remove punctuation
tokens = tokens.apply(lambda words: map(partial(re.sub, "&lt;/?.*?&gt;", " &lt;&gt;
    ↪ "), words))
tokens = tokens.apply(lambda words: map(partial(re.sub, "<.*?>", ""), words))
tokens = tokens.apply(lambda words: map(partial(re.sub, "(\\d|\\W)+", " "), words))
tokens = tokens.apply(lambda words: [word.strip() for word in words if word.strip()
    ↪ ])

# Lemmatize
lem = WordNetLemmatizer()
tokens = tokens.apply(lambda words: list(map(lem.lemmatize, words)))

# Stem
porter_stemmer = PorterStemmer()
tokens = tokens.apply(lambda words: list(map(porter_stemmer.stem, words)))

# Join
clean_text = tokens.apply(" ".join)

# Count frequencies
vectorizer = CountVectorizer()
vectorizer.fit(clean_text)
```

```
df[out] = vectorizer.transform(clean_text)
```

37. Newton message: [The columns here depends on what the user selected in the previous form as categorical columns]

```
Please, select which categorical columns do you want to encode and click on '
    ↪ Continue'
####form#:
-check#:[categorical column]#:*yes
-submit#:Continue
####fol#:
- Which type of preprocessing?
```

37.A. Newton-message: Sample 1

```
Please, select which categorical columns do you want to encode and click on '
    ↪ Continue'
####form#:
-check#:anaemia#:*yes
-check#:diabetes#:*yes
-check#:high_blood_pressure#:*yes
-check#:sex#:*yes
-check#:smoking#:*yes
-check#:DEATH_EVENT#:*yes
-submit#:Continue
####fol#:
- What is encoding?
```

37.B. Newton-message: Sample 2

```
Please, select which categorical columns do you want to encode and click on '
    ↪ Continue'
####form#:
-check#:Sex#:*yes
-check#:ChestPainType#:*yes
-check#:RestingECG#:*yes
-check#:ExerciseAngina#:*yes
-check#:ST_Slope#:*yes
-submit#:Continue
####fol#:
- What is encoding?
```

>>> User enactable suggestion: What is encoding?

Go to 38

>>> User enactable suggestion: Submit

Set [columns]

Go to 39

38. Newton message:

```
Encoding is an operation that converts a finite set of textual categories into
    ↪ integers ranging from 0 to the number of categories
```

39. Newton message:

```
Copy the following code to the notebook:
####code#:
from sklearn.preprocessing import LabelEncoder
columns = ['[columns1]', '[columns2]']
for column in columns:
    encoder = LabelEncoder()
    df[column] = encoder.fit_transform(df[column])
df.info()
```

39.A. Newton-message: Sample 1

```
Copy the following code to the notebook:
####code#:
from sklearn.preprocessing import LabelEncoder
columns = ['anaemia', 'diabetes', 'high_blood_pressure', 'sex', 'smoking'
```

```
, 'DEATH_EVENT'
]
for column in columns:
    encoder = LabelEncoder()
    df[column] = encoder.fit_transform(df[column])
df.info()
```

39.B. Newton-message: Sample 2

```
Copy the following code to the notebook:
####code#:
from sklearn.preprocessing import LabelEncoder
columns = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope'
]
for column in columns:
    encoder = LabelEncoder()
    df[column] = encoder.fit_transform(df[column])
df.info()
```

40. Newton message: [Context: clean text and encode depend on the existing of these operations in the source]

```
Which operation do you want to know about?
####fol#:
- Check for Null values
- Check for Invalid Zero values
- Clean text values
- Encode Categorical values
```

>>> User enactable suggestion: What is Check for Null values?

Go to 17

>>> User enactable suggestion: Check for Invalid Zero values

Go to 25

>>> User enactable suggestion: Clean text values

Go to 28

>>> User enactable suggestion: Encode Categorical values

Go to 38

41. Newton message:

```
Which dataframe do you want to split?
```

>>> User reply: [dataframe=df]

Set [dataframe]

Go to 42

42. Newton message:

```
Which column do you want to classify?
```

>>> User reply: [classification_column]

Set [classification_column]

Go to 43

43. Newton message:

```
Copy the following code to the notebook:
####code#:
# split data for training
x = df.drop('[classification_column]', axis=1)
y = df['[classification_column]']
####fol#:
- Continue
- Read 'drop' Documentation
```

43.A. Newton-message: Sample 1

```
Copy the following code to the notebook:
####code#:
# split data for training
```

```
x = df.drop('DEATH_EVENT', axis=1)
y = df['DEATH_EVENT']
####fol#:
- Continue
- Read 'drop' Documentation
```

43.B. Newton-message: Sample 2

```
Copy the following code to the notebook:
####code#:
# split data for training
x = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']
####fol#:
- Continue
- Read 'drop' Documentation
```

>>> User enactable suggestion: Continue

Go to 44

>>> User enactable suggestion: Read 'drop' Documentation

Go to 53

44. Newton message: [Set context X_train, Y_train]

```
Now, copy the following code to the notebook:
####code#:
# Separate and train the data to be ready for implementation
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
    ↪ random_state=21)
####fol#:
- Continue
- Read 'train_test_split' Documentation
```

>>> User enactable suggestion: Continue

Go to 45

>>> User enactable suggestion: Read 'train_test_split' Documentation

Go to 52

45. Newton message:

```
It is often good to scale the data before building the classification model. Do you
    ↪ want to do it?
####fol#:
- Yes
- No
```

>>> User enactable suggestion: Yes

Go to 46

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

46. Newton message: [Context: has continue]

```
Copy the following code to the notebook:
####code#:
# Fully format and organize trained data
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
std_scaler.fit(x)
X_train_std = std_scaler.transform(X_train)
X_test_std = std_scaler.transform(X_test)

X_train = pd.DataFrame(X_train_std, columns=x.columns)
X_test = pd.DataFrame(X_test_std, columns=x.columns)

# Show trained and organized data
display(X_train.head())
```

```
display(X_test.head())
####fol#:
- Continue
- Read 'StandardScaler' Documentation
- Read 'transform' Documentation
- Read 'DataFrame' Documentation
- Read 'head' Documentation
- Read 'display' Documentation
```

46.A. Newton-message: [Context: no continue]

```
Copy the following code to the notebook:
####code#:
# Fully format and organize trained data
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
std_scaler.fit(x)
X_train_std = std_scaler.transform(X_train)
X_test_std = std_scaler.transform(X_test)

X_train = pd.DataFrame(X_train_std, columns=x.columns)
X_test = pd.DataFrame(X_test_std, columns=x.columns)

# Show trained and organized data
display(X_train.head())
display(X_test.head())
####fol#:
- Read 'StandardScaler' Documentation
- Read 'transform' Documentation
- Read 'DataFrame' Documentation
- Read 'head' Documentation
- Read 'display' Documentation
```

>>> User enactable suggestion: Continue

Continue to next step.

>>> User enactable suggestion: Read 'StandardScaler' Documentation

Go to 47

>>> User enactable suggestion: Read 'transform' Documentation

Go to 48

>>> User enactable suggestion: Read 'DataFrame' Documentation

Go to 49

>>> User enactable suggestion: Read 'head' Documentation

Go to 50

>>> User enactable suggestion: Read 'display' Documentation

Go to 51

47. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.
    ↪ StandardScaler.html
```

48. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.
    ↪ StandardScaler.html#sklearn.preprocessing.StandardScaler.transform
```

49. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
```

50. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html
```

51. Newton message:

    ####web-panel#:
    https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html

52. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.
    ↪ train_test_split.html

53. Newton message:

    ####web-panel#:
    https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop.html

54. Newton message:

    What is your X an Y train data?

>>> User reply: [X_train and Y_train]

    Go to 55

55. Newton message:

    Which algorithm do you want to use for the classification?
    ####fol#:
    - Logistic Regression
    - KNN
    - Decision Tree
    - Random Forest

>>> User enactable suggestion: Logistic Regression

    Go to 56

>>> User enactable suggestion: KNN

    Go to 67

>>> User enactable suggestion: Decision Tree

    Go to 78

>>> User enactable suggestion: Random Forest

    Go to 89

56. Newton message:

    Copy the following code to the notebook:
    ####code#:
    ```
    # Logistic Regression
    from sklearn.linear_model import LogisticRegression
    lr = LogisticRegression(C = 80)
    lr.fit(X_train, y_train)
    print(f"Score: {lr.score(X_train , y_train)}")
    ```
    ####text#:
    Do you want to validate it?
    ####fol#:
    - Yes
    - No
    - Read 'LogisticRegression' Documentation
    - Read 'fit' Documentation
    - Read 'score' Documentation

>>> User enactable suggestion: Yes

    Go to 57 [Context: If not split]

    Go to 58 [Otherwise]

>>> User enactable suggestion: No

    Continue to the next step if available. Otherwise, do not reply.

>>> User enactable suggestion: Read 'LogisticRegression' Documentation

    Go to 64

>>> User enactable suggestion: Read 'fit' Documentation

Go to 65

>>> User enactable suggestion: Read 'score' Documentation

Go to 66

57. Newton message:

```
What is your X and Y test set?
```

>>> User reply: [X_test and y_test]

Go to 58

58. Newton message:

```
Copy the following code to the notebook:
####code#:
# Predict test data
Y_predlr = lr.predict(X_test)
# Compare classification predictions to actual values
data = pd.DataFrame({"Y_test" : y_test , "Y_predlr": Y_predlr})
data.head(10)
####fol#:
- Calculate Precision, Recall, F1-score, Support
- Read 'predict' Documentation
- Read 'DataFrame' Documentation
- Read 'head' Documentation
```

>>> User reply: Calculate Precision, Recall, F1-score, Support

Go to 59

>>> User enactable suggestion: Read 'predict' Documentation

Go to 61

>>> User enactable suggestion: Read 'DataFrame' Documentation

Go to 62

>>> User enactable suggestion: Read 'head' Documentation

Go to 63

59. Newton message:

```
Copy the following code to the notebook:
####code#:
# Show information about classification performance
from sklearn.metrics import classification_report
print(classification_report(y_test, Y_predlr))
####fol#:
- Read 'classification_report' Documentation
```

>>> User enactable suggestion: Read 'classification_report' Documentation

Go to 60

60. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
    ↪ classification_report.html
```

61. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.
    ↪ LogisticRegression.html#sklearn.linear_model.LogisticRegression.predict
```

62. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
```

63. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html
```

64. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.
    ↪ LogisticRegression.html

65. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.
    ↪ LogisticRegression.html#sklearn.linear_model.LogisticRegression.fit

66. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.
    ↪ LogisticRegression.html#sklearn.linear_model.LogisticRegression.score

67. Newton message:

    Copy the following code to the notebook:
    ####code#:
    #K-Nearest Neighbor
    from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors=10)
    knn.fit(X_train , y_train)
    print(f"Score: {knn.score(X_train , y_train)}")
    ####text#:
    Do you want to validate it?
    ####fol#:
    - Yes
    - No
    - Read 'KNeighborsClassifier' Documentation
    - Read 'fit' Documentation
    - Read 'score' Documentation

>>> User enactable suggestion: Yes

    Go to 68 [Context: If not split]

    Go to 69 [Otherwise]

>>> User enactable suggestion: No

    Continue to the next step if available. Otherwise, do not reply.

>>> User enactable suggestion: Read 'KNeighborsClassifier' Documentation

    Go to 75

>>> User enactable suggestion: Read 'fit' Documentation

    Go to 76

>>> User enactable suggestion: Read 'score' Documentation

    Go to 77

68. Newton message:

    What is your X and Y test set?

>>> User reply: [X_test and y_test]

    Go to 69

69. Newton message:

    Copy the following code to the notebook:
    ####code#:
    # Predict test data
    Y_predknn = knn.predict(X_test)
    # compare classification predictions to actual values
    data = pd.DataFrame({"Y_test" : y_test , "Y_predknn": Y_predknn})
    data.head(10)
    ####fol#:
    - Calculate Precision, Recall, F1-score, Support
    - Read 'predict' Documentation
    - Read 'DataFrame' Documentation
    - Read 'head' Documentation

>>> User reply: Calculate Precision, Recall, F1-score, Support

Go to 70

>>> User enactable suggestion: Read 'predict' Documentation

Go to 72

>>> User enactable suggestion: Read 'DataFrame' Documentation

Go to 73

>>> User enactable suggestion: Read 'head' Documentation

Go to 74

70. Newton message:

```
Copy the following code to the notebook:
####code#:
# Show information about classification performance
from sklearn.metrics import classification_report
print(classification_report(y_test, Y_predknn))
####fol#:
- Read 'classification_report' Documentation
```

>>> User enactable suggestion: Read 'classification_report' Documentation

Go to 71

71. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
    ↪ classification_report.html
```

72. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.
    ↪ KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.predict
```

73. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
```

74. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html
```

75. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.
    ↪ KNeighborsClassifier.html
```

76. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.
    ↪ KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.fit
```

77. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.
    ↪ KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier.score
```

78. Newton message:

```
Copy the following code to the notebook:
####code#:
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_features=9 , max_depth=4)
dt.fit(X_train , y_train)
print(f"Score: {dt.score(X_train , y_train)}")
```

```
####text#:
Do you want to validate it?
####fol#:
- Yes
- No
- Read 'DecisionTreeClassifier' Documentation
- Read 'fit' Documentation
- Read 'score' Documentation
```

>>> User enactable suggestion: Yes

Go to 79 [Context: If not split]

Go to 80 [Otherwise]

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

>>> User enactable suggestion: Read 'DecisionTreeClassifier' Documentation

Go to 86

>>> User enactable suggestion: Read 'fit' Documentation

Go to 87

>>> User enactable suggestion: Read 'score' Documentation

Go to 88

79. Newton message:

```
What is your X and Y test set?
```

>>> User reply: [X_test and y_test]

Go to 80

80. Newton message:

```
Copy the following code to the notebook:
####code#:
# Predict test data
Y_preddt = dt.predict(X_test)
# compare classification predictions to actual values
data = pd.DataFrame({"Y_test" : y_test , "Y_pred": Y_preddt})
data.head(10)
####fol#:
- Calculate Precision, Recall, F1-score, Support
- Read 'predict' Documentation
- Read 'DataFrame' Documentation
- Read 'head' Documentation
```

>>> User reply: Calculate Precision, Recall, F1-score, Support

Go to 81

>>> User enactable suggestion: Read 'predict' Documentation

Go to 83

>>> User enactable suggestion: Read 'DataFrame' Documentation

Go to 84

>>> User enactable suggestion: Read 'head' Documentation

Go to 85

81. Newton message:

```
Copy the following code to the notebook:
####code#:
# Show information about classification performance
from sklearn.metrics import classification_report
print(classification_report(y_test, Y_preddt))
####fol#:
- Read 'classification_report' Documentation
```

>>> User enactable suggestion: Read 'classification_report' Documentation

Go to 82

82. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
        ↪ classification_report.html

83. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.tree.
        ↪ DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.predict

84. Newton message:

    ####web-panel#:
    https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

85. Newton message:

    ####web-panel#:
    https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html

86. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.tree.
        ↪ DecisionTreeClassifier.html

87. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.tree.
        ↪ DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.fit

88. Newton message:

    ####web-panel#:
    https://scikit-learn.org/stable/modules/generated/sklearn.tree.
        ↪ DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.score

89. Newton message:

    Copy the following code to the notebook:
    ####code#:
    ```
    # Random Forest
    from sklearn.ensemble import RandomForestClassifier
    rf = RandomForestClassifier(n_estimators=4 , max_depth=3 , min_samples_split=25 ,
        ↪ max_features=4)
    rf.fit(X_train, y_train)
    print(f"Score: {rf.score(X_train, y_train)}")
    ```
    ####text#:
    Do you want to validate it?
    ####fol#:
    - Yes
    - No
    - Read 'RandomForestClassifier' Documentation
    - Read 'fit' Documentation
    - Read 'score' Documentation

>>> User enactable suggestion: Yes

Go to 90 [Context: If not split]

Go to 91 [Otherwise]

>>> User enactable suggestion: No

Continue to the next step if available. Otherwise, do not reply.

>>> User enactable suggestion: Read 'RandomForestClassifier' Documentation

Go to 97

>>> User enactable suggestion: Read 'fit' Documentation

Go to 98

>>> User enactable suggestion: Read 'score' Documentation

Go to 99

90. Newton message:

```
What is your X and Y test set?
```

>>> User reply: [X_test and y_test]

Go to 91

91. Newton message:

```
Copy the following code to the notebook:
####code#:
# Predict test data
Y_predrf = rf.predict(X_test)
# compare classification predictions to actual values
data = pd.DataFrame({"Y_test" : y_test , "Y_pred": Y_predrf})
data.head(10)
####fol#:
- Calculate Precision, Recall, F1-score, Support
- Read 'predict' Documentation
- Read 'DataFrame' Documentation
- Read 'head' Documentation
```

>>> User reply: Calculate Precision, Recall, F1-score, Support

Go to 92

>>> User enactable suggestion: Read 'predict' Documentation

Go to 94

>>> User enactable suggestion: Read 'DataFrame' Documentation

Go to 95

>>> User enactable suggestion: Read 'head' Documentation

Go to 96

92. Newton message:

```
Copy the following code to the notebook:
####code#:
# Show information about classification performance
from sklearn.metrics import classification_report
print(classification_report(y_test, Y_predrf))
####fol#:
- Read 'classification_report' Documentation
```

>>> User enactable suggestion: Read 'classification_report' Documentation

Go to 93

93. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.
    ↪ classification_report.html
```

94. Newton message:

```
####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.
    ↪ RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.predict
```

95. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
```

96. Newton message:

```
####web-panel#:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.head.html
```

97. Newton message:

####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.
↪ RandomForestClassifier.html

98. Newton message:

####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.
↪ RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.fit

99. Newton message:

####web-panel#:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.
↪ RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.score

100. Newton message:

Which step do you want to understand?
####fol#:
- Load data
- Preprocess the data
- Split test and train set
- Build model
- Validate model

>>> User enactable suggestion: Load data

Go to 101

>>> User enactable suggestion: Preprocess the data

Go to 102

>>> User enactable suggestion: Split test and train set

Go to 103

>>> User enactable suggestion: Build model

Go to 104

>>> User enactable suggestion: Validate model

Go to 105

101. Newton message:

In this step, you load the data from a data source to use in the analysis

102. Newton message:

In this step, you preprocess the data to remove null values, clean and encode
↪ categorical variables, remove improper outliers from numerical variables, and
↪ filter out columns that are not necessary for the data analysis

103. Newton message:

In this step, you divide the dataset into a train set and a test set. The train set
↪ will be used to train the classifier and the test set will be used to validate
↪ the trained model and extract metrics

104. Newton message:

In this step, you build the classifier based on the training data

105. Newton message:

In this step, you validate the classification model using the test data

2.1 Warmup

>>> User query: "Generate normal distribution"

Go to 106

106. Newton message:

```
To generate a normal distribution, use the following code:
####code#:
import numpy as np
mu, sigma = 0, 0.1
s = np.random.normal(mu, sigma, 1000)
####fol#:
- Read 'normal' Documentation
```

>>> User enactable suggestion: Read 'normal' Documentation

Go to 107

107. Newton message:

```
####web-panel#:
https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html
```

>>> User query: "read documentation about plotting"

Go to 108

108. Newton message:

```
Here is the documentation for plotting a histogram:
####web-panel#:
https://seaborn.pydata.org/generated/seaborn.histplot.html
```

>>> User query: "plot the distribution"

Go to 109

109. Newton message:

```
Copy the following code to the notebook:
####code#:
import seaborn as sns
sns.histplot(s)
####fol#:
- Read 'histplot' documentation
```

>>> User enactable suggestion: Read 'histplot' Documentation

Go to 110

110. Newton message:

```
####web-panel#:
https://seaborn.pydata.org/generated/seaborn.histplot.html
```