

Theory Activity No. 1

Name : Rohan Ajay Burande

PRN : 202401040188

Div : CS1

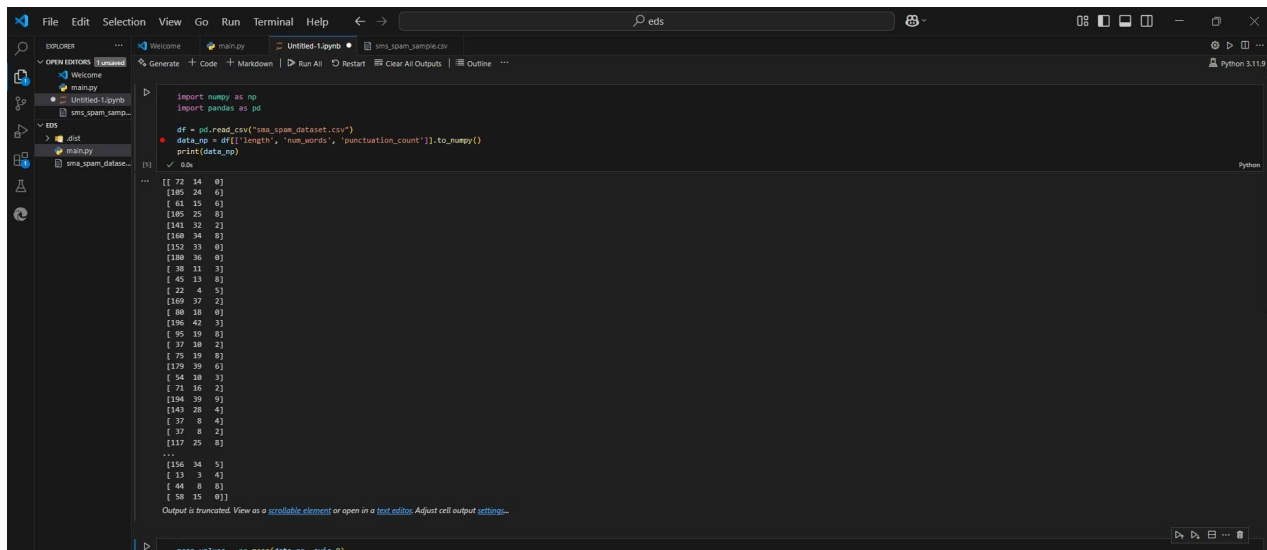
Roll No : CS1-78

Data Set : SMS SPAM COLLECTION

NumPy:-

1. Convert DataFrame columns to NumPy array

solution:



The screenshot shows a Jupyter Notebook interface with a file explorer on the left and a code editor on the right. The code editor contains the following Python code:

```
import numpy as np
import pandas as pd

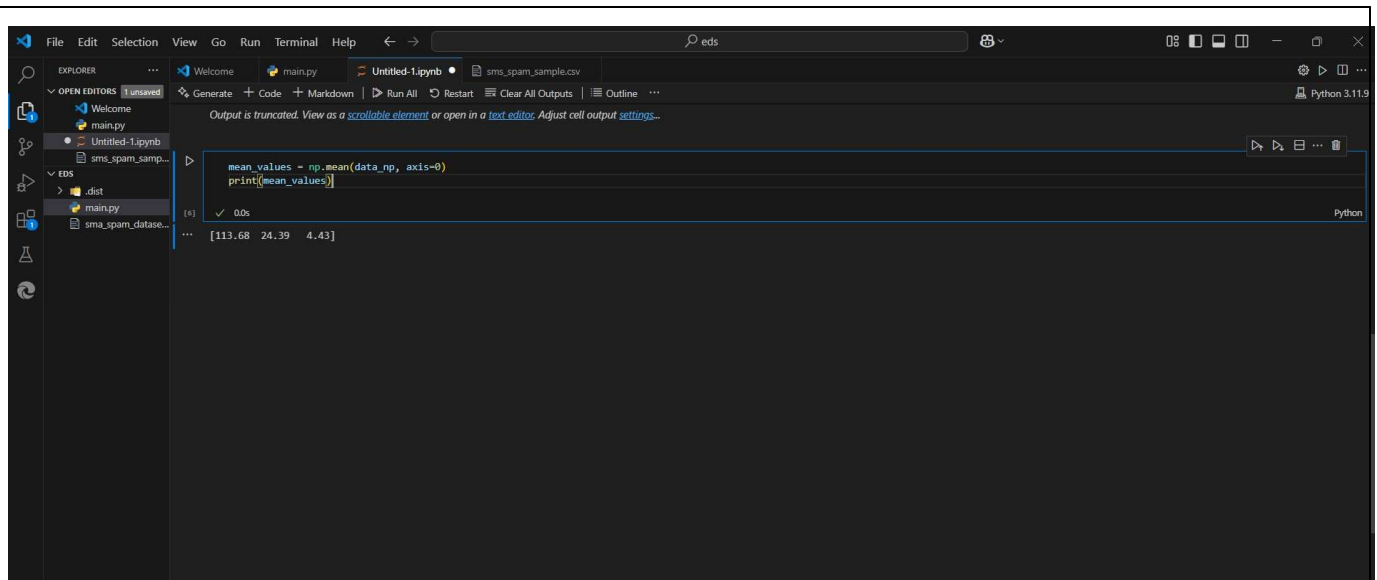
df = pd.read_csv("sms_spam_dataset.csv")
data_np = df[['length', 'num_words', 'punctuation_count']].to_numpy()
print(data_np)
```

The output of the code is displayed below the code cell, showing a truncated view of the NumPy array:

```
[[ 72 14  0]
 [185 24  6]
 [ 63 15  6]
 [185 25  8]
 [141 32  2]
 [168 34  8]
 [152 33  0]
 [188 36  0]
 [ 38 11  3]
 [ 45 13  5]
 [ 22  4  5]
 [189 37  2]
 [ 88 38  0]
 [196 42  3]
 [ 95 19  8]
 [ 37 28  2]
 [ 75 19  8]
 [179 39  6]
 [ 54 18  2]
 [ 71 16  2]
 [194 39  9]
 [143 28  4]
 [ 37  8  4]
 [ 37  8  2]
 [117 25  8]
 ...
 [156 34  5]
 [ 15  3  4]
 [ 44  8  8]
 [ 58 15  0]]
```

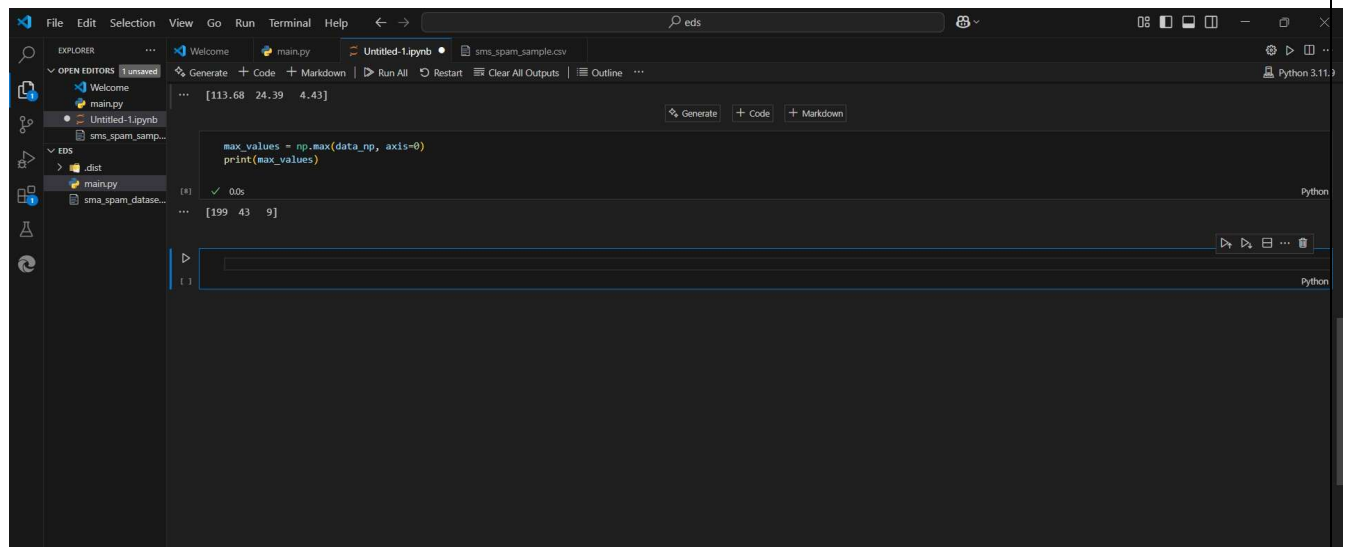
The output is truncated, and a message at the bottom indicates: "Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings--".

2. Calculate mean of each numeric column



3. Find max values in each column

solution:



4. Find rows where message length is above average

solution:

```
above_avg = data_np[:, 0] > np.mean(data_np[:, 0])
print(df[above_avg])
```

	label	message	length	num_words	punctuation_count
4	ham	Sample message 4	141	32	2
5	spam	Sample message 5	160	34	8
6	ham	Sample message 6	152	33	0
7	ham	Sample message 7	180	36	0
11	ham	Sample message 11	169	37	2
...
181	ham	Sample message 181	167	37	3
183	ham	Sample message 183	118	27	0
185	ham	Sample message 185	191	38	1
193	ham	Sample message 193	120	26	2
196	spam	Sample message 196	156	34	5

[108 rows x 5 columns]

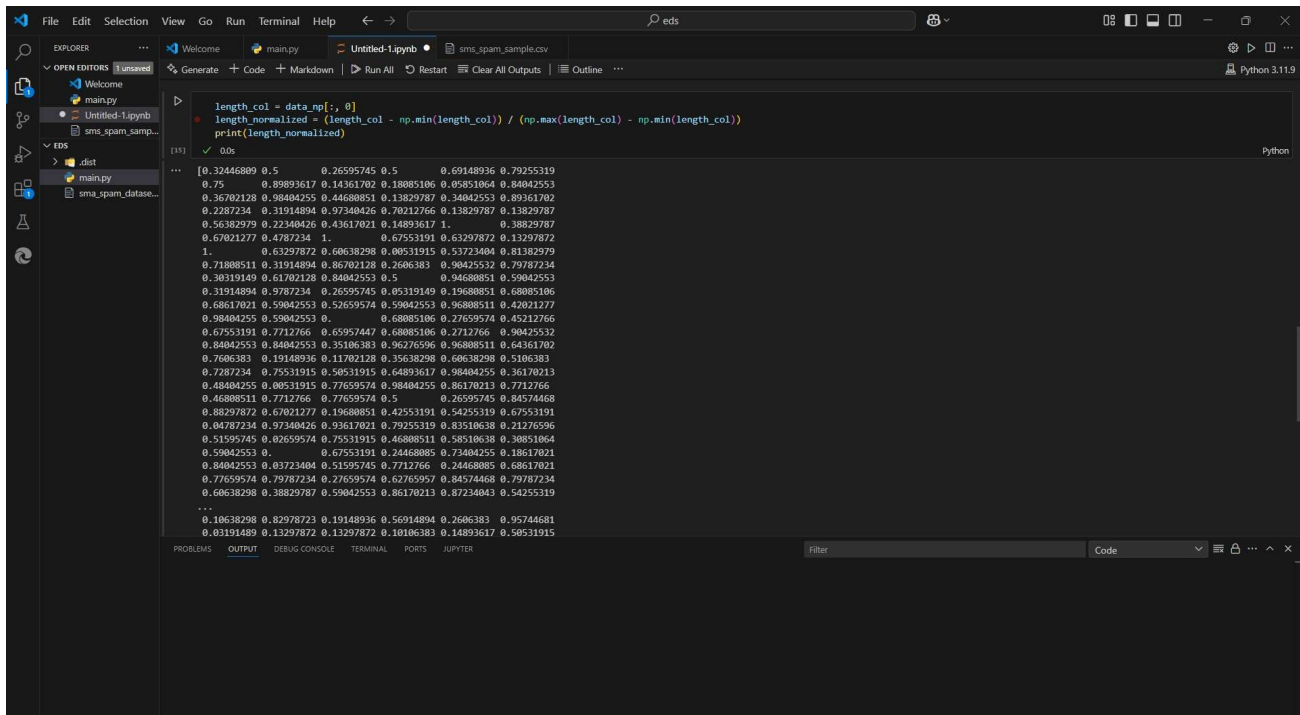
5. Count messages with more than 5 punctuation marks

solution:

```
count_punct = np.sum(data_np[:, 2] > 5)
print(count_punct)
```

74

6. Normalize the 'length' column



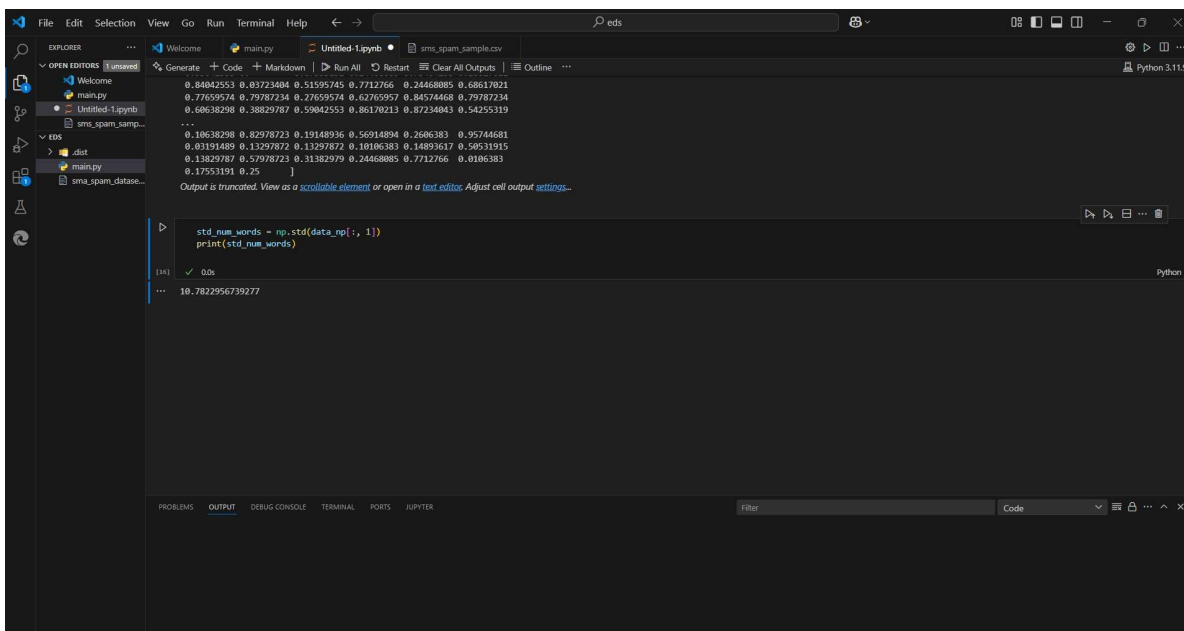
The screenshot shows a Jupyter Notebook interface with a file explorer on the left. The main area displays a code cell with the following Python code:

```
length_col = data_np[:, 0]
length_normalized = (length_col - np.min(length_col)) / (np.max(length_col) - np.min(length_col))
print(length_normalized)
```

The output of the code is a large array of normalized values, starting with `[0.32446809 0.5 0.26595745 0.5 0.69148936 0.79255319` and ending with `0.03191489 0.13297872 0.13297872 0.10106383 0.14893617 0.50531915`. The output is truncated at the bottom.

7. Compute standard deviation of 'num_words'

solution:



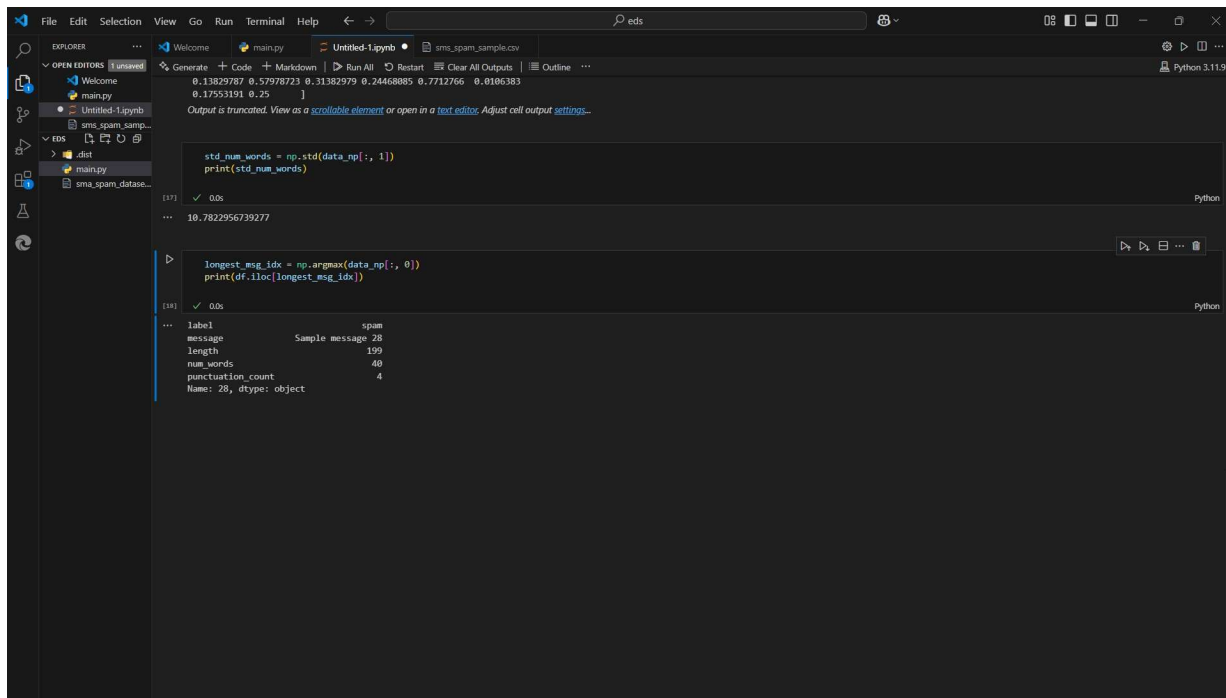
The screenshot shows a Jupyter Notebook interface with a file explorer on the left. The main area displays a code cell with the following Python code:

```
std_num_words = np.std(data_np[:, 1])
print(std_num_words)
```

The output of the code is a single value: `10.7822956739277`. The output is truncated at the bottom.

8. Find index of longest message.

solution:



The screenshot shows a Jupyter Notebook with two code cells. The first cell calculates the standard deviation of the number of words in each message. The second cell finds the index of the longest message and prints its details.

```
std_num_words = np.std(data_np[:, 1])
print(std_num_words)
```

Output:

```
0.13829787 0.57978723 0.31382979 0.24468885 0.7712766 0.0186383
0.17553191 0.25 ]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
longest_msg_idx = np.argmax(data_np[:, 0])
print(df.iloc[longest_msg_idx])
```

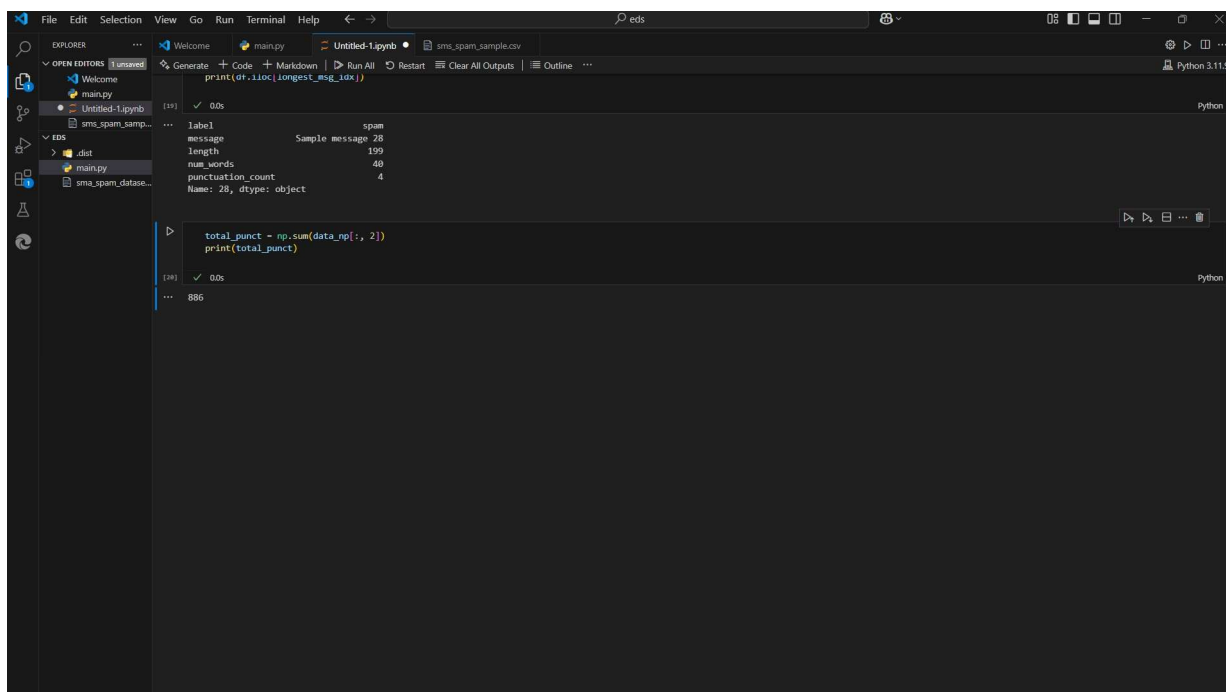
Output:

	spam
label	Sample message 28
message	
length	199
num_words	48
punctuation_count	4

Name: 28, dtype: object

9. Sum of punctuation across all messages

solution:



The screenshot shows a Jupyter Notebook with two code cells. The first cell finds the index of the longest message. The second cell calculates the sum of punctuation across all messages.

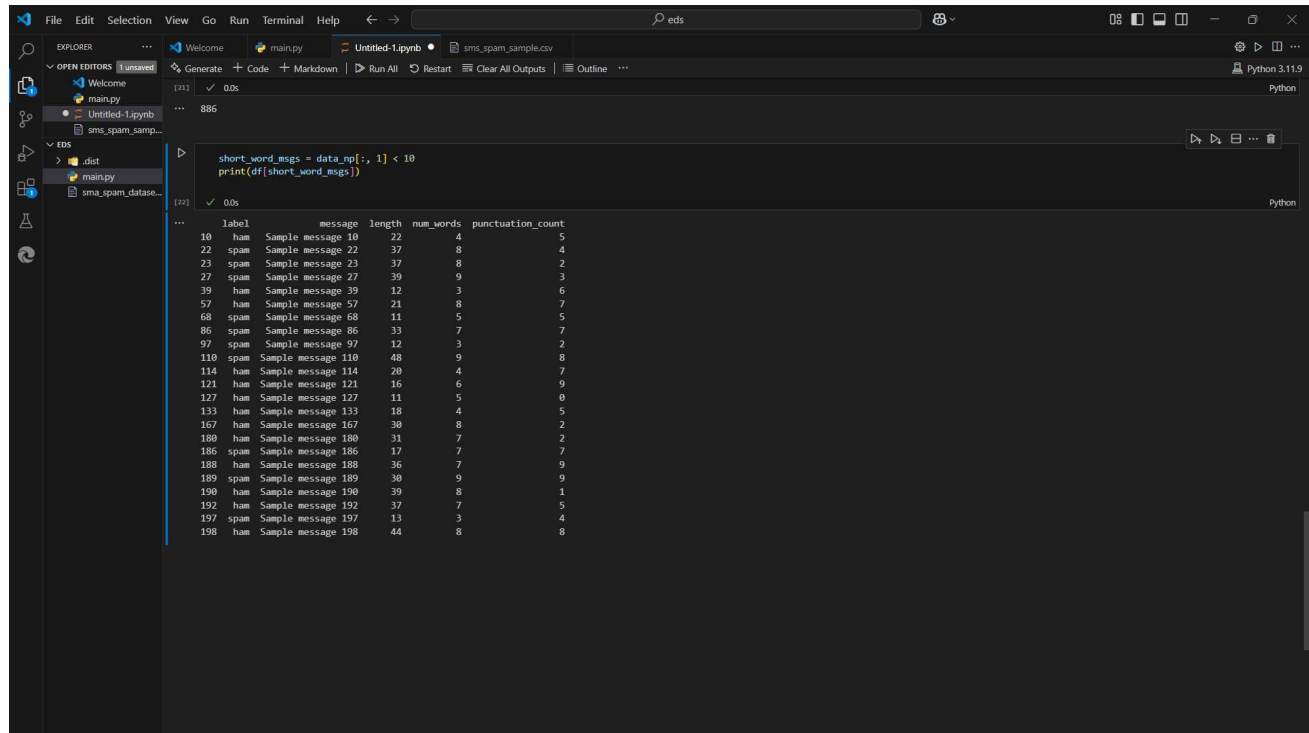
```
total_punct = np.sum(data_np[:, 2])
print(total_punct)
```

Output:

```
886
```

10. Messages where number of words is less than 10

solution:



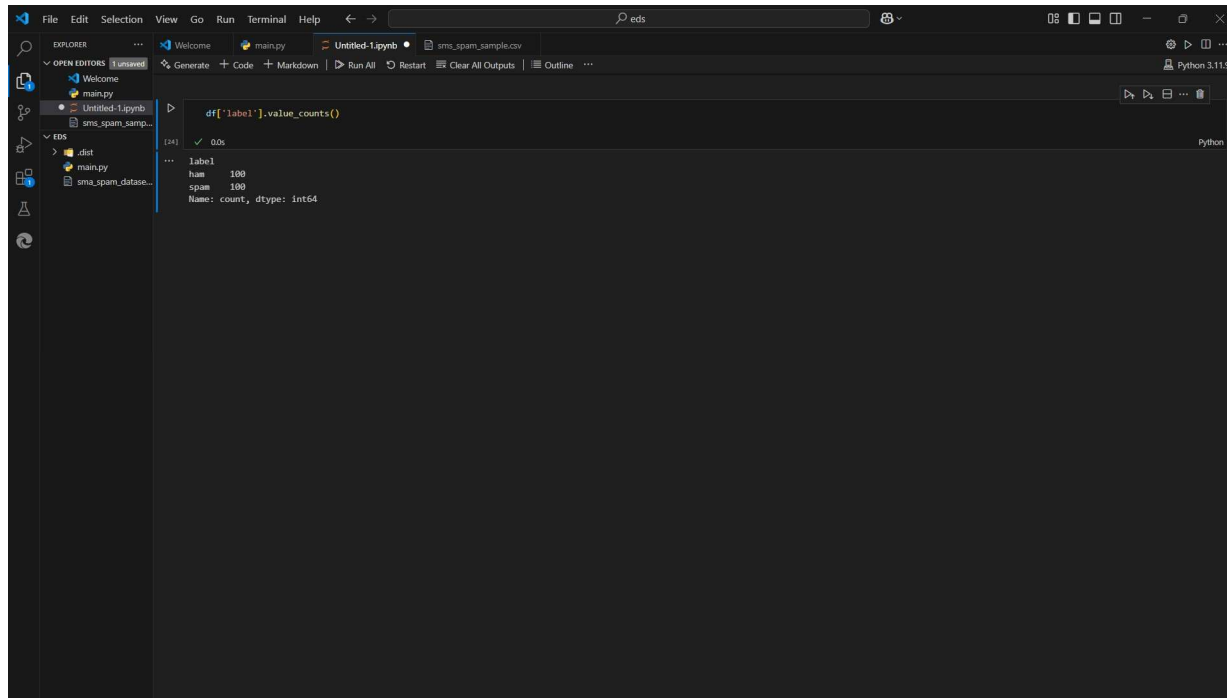
```
short_word_msgs = data_df[ num_words < 10 ]
print(df[short_word_msgs])
```

	label	message	length	num_words	punctuation_count
10	ham	Sample message 10	22	4	5
22	spam	Sample message 22	37	8	4
23	spam	Sample message 23	37	8	2
27	spam	Sample message 27	39	9	3
39	ham	Sample message 39	12	3	6
57	ham	Sample message 57	21	8	7
68	spam	Sample message 68	11	5	5
86	spam	Sample message 86	33	7	7
97	spam	Sample message 97	12	3	2
110	spam	Sample message 110	48	9	8
114	ham	Sample message 114	20	4	7
121	ham	Sample message 121	16	6	9
127	ham	Sample message 127	11	5	0
133	ham	Sample message 133	18	4	5
167	ham	Sample message 167	30	8	2
180	ham	Sample message 180	31	7	2
186	spam	Sample message 186	17	7	7
188	ham	Sample message 188	36	7	9
189	spam	Sample message 189	30	9	9
190	ham	Sample message 190	39	8	1
192	ham	Sample message 192	37	7	5
197	spam	Sample message 197	13	3	4
198	ham	Sample message 198	44	8	8

pandas:

1. How many spam and ham messages are there?

solution:



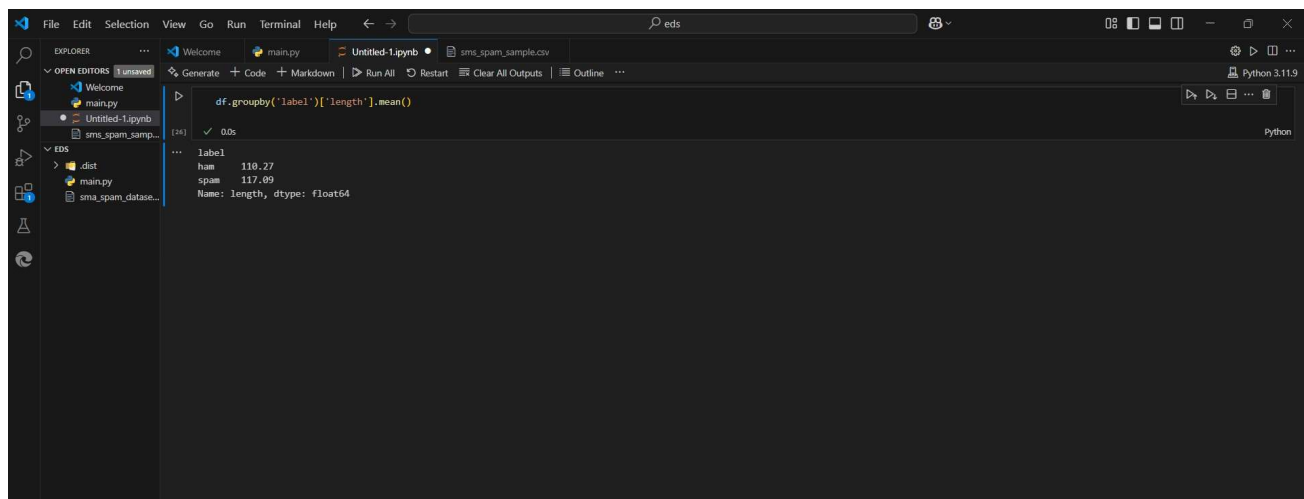
```
df['label'].value_counts()
```

```
[14] ✓ 0.0s
```

```
label
ham    100
spam   100
Name: count, dtype: int64
```

2. What is the average length of spam and ham messages?.

solution:



```
df.groupby('label')['length'].mean()
```

```
[14] ✓ 0.0s
```

```
label
ham    110.27
spam   117.09
Name: length, dtype: float64
```


3. Which message has the most punctuation?

solution:

The screenshot shows the JupyterLab interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The Explorer panel on the left shows the file structure with 'main.py' and 'sms_spam_sample.csv' selected. The main editor area displays a Jupyter Notebook with a pandas DataFrame and a console output.

The Jupyter Notebook code is as follows:

```
[27]: 0.0s
...
label
ham    118.27
spam   117.89
Name: length, dtype: float64

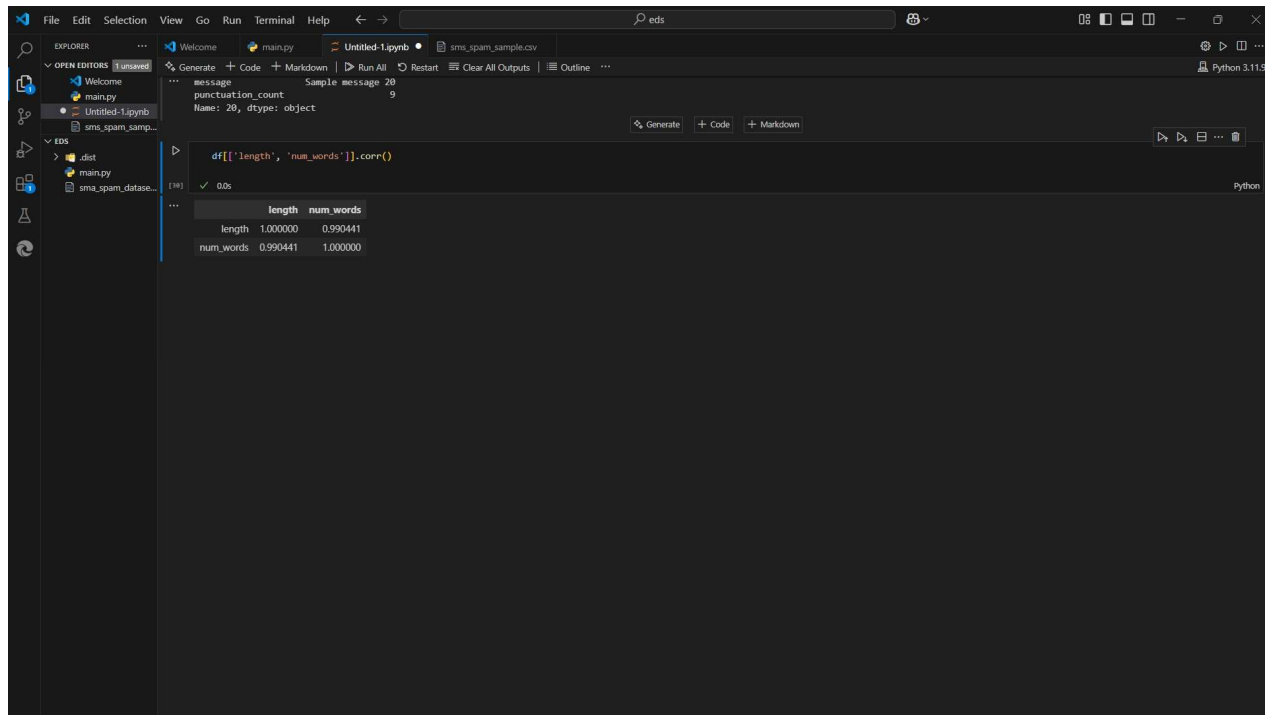
df.loc[df['punctuation_count'].idxmax()][['message', 'punctuation_count']]

[28]: 0.0s
...
message      Sample message 20
punctuation_count          9
Name: 20, dtype: object
```

The console output shows the result of the last cell execution, displaying the message and punctuation count for the sample message 20.

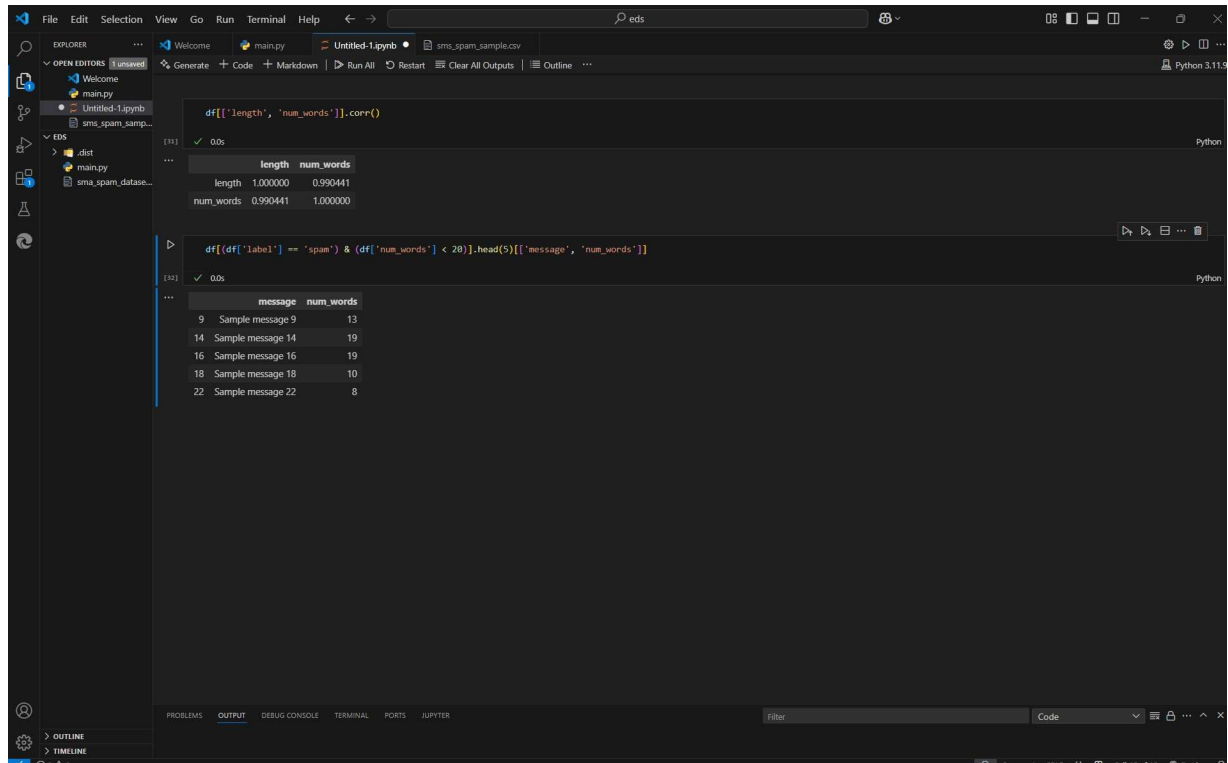
4. What is the correlation between message length and number of words?

solution:



5. What are the first 5 spam messages with less than 20 words?

solution:



```
df[['length', 'num_words']].corr()
```

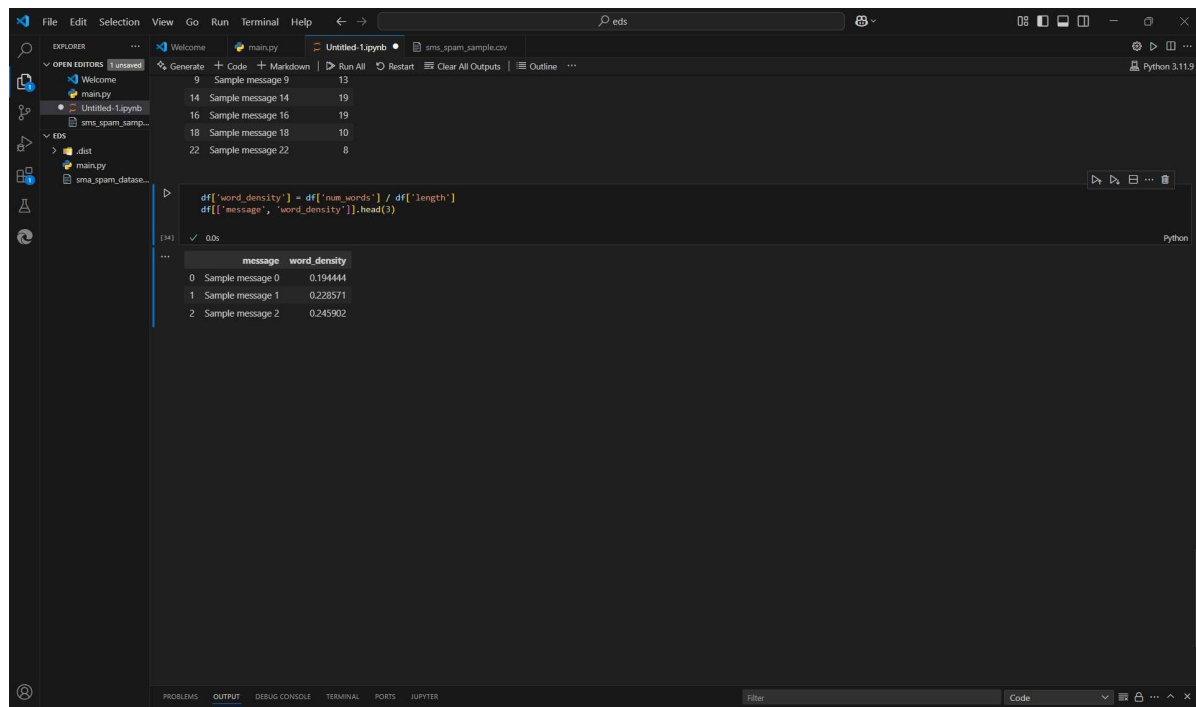
	length	num_words
length	1.000000	0.990441
num_words	0.990441	1.000000

```
df[(df['label'] == 'spam') & (df['num_words'] < 20)].head(5)[['message', 'num_words']]
```

	message	num_words
9	Sample message 9	13
14	Sample message 14	19
16	Sample message 16	19
18	Sample message 18	10
22	Sample message 22	8

6. Add a column for word density (words per character) and show the top 3 rows

solution:

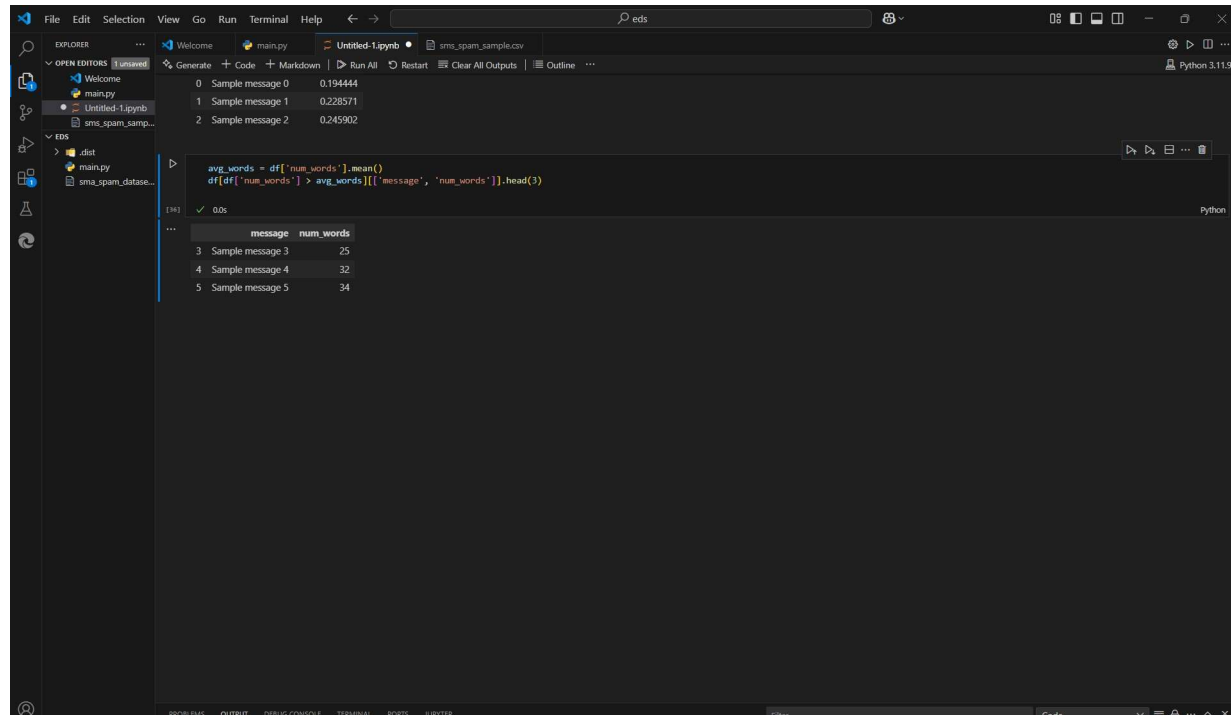


```
df['word_density'] = df['num_words'] / df['length']
df[['message', 'word_density']].head(3)
```

	message	word_density
0	Sample message 0	0.194444
1	Sample message 1	0.228571
2	Sample message 2	0.245902

7. What are the messages with word counts above the average?

solution:



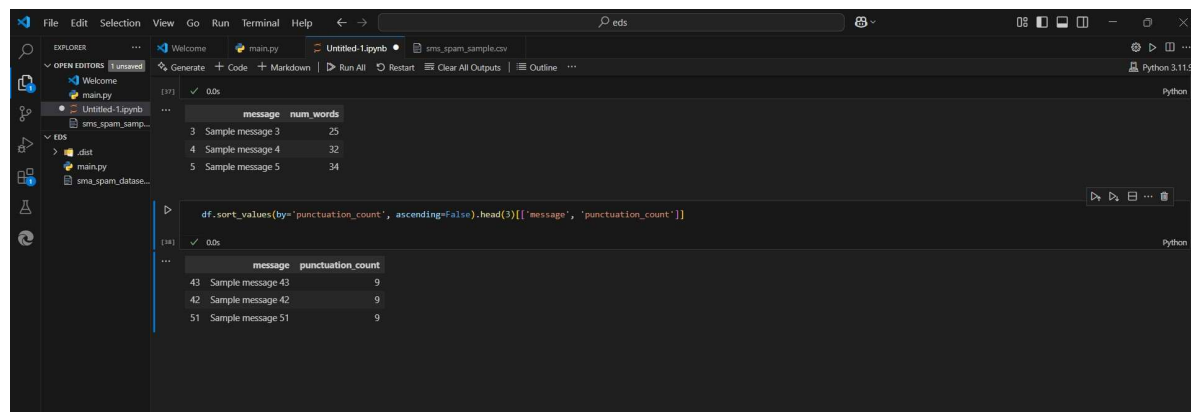
The screenshot shows a JupyterLab environment with a file explorer on the left and a code editor in the center. The code editor contains a Python script that calculates the average word count and filters messages with word counts above the average. The output of the script is displayed in a table.

```
avg_words = df['num_words'].mean()
df[df['num_words'] > avg_words][['message', 'num_words']].head(3)
```

	message	num_words
3	Sample message 3	25
4	Sample message 4	32
5	Sample message 5	34

8. Sort the messages by punctuation count and show the top 3

solution:



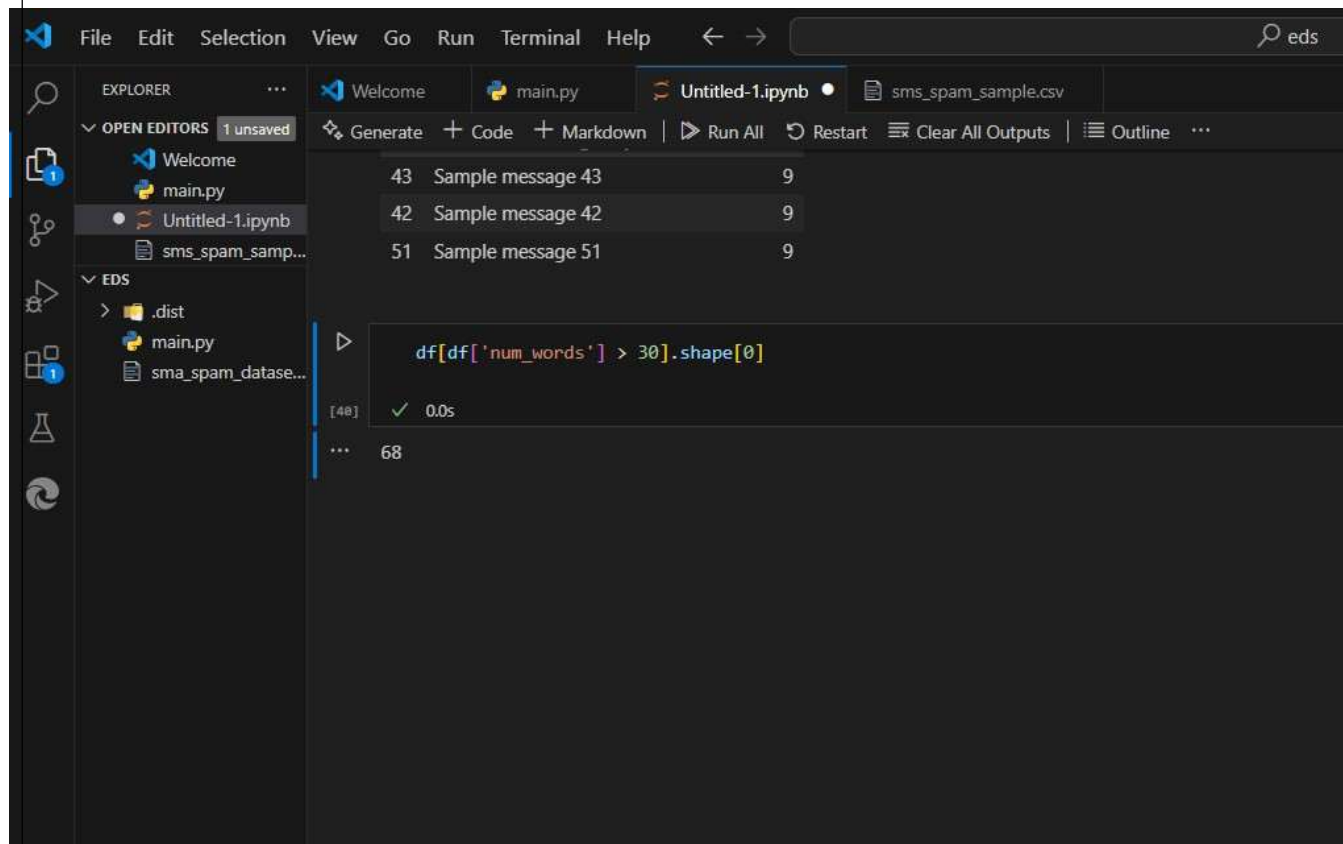
The screenshot shows a JupyterLab environment with a file explorer on the left and a code editor in the center. The code editor contains a Python script that sorts messages by punctuation count and displays the top 3. The output of the script is displayed in a table.

```
df.sort_values(by='punctuation_count', ascending=False).head(3)[['message', 'punctuation_count']]
```

	message	punctuation_count
43	Sample message 43	9
42	Sample message 42	9
51	Sample message 51	9

9. Count how many messages have more than 30 words

solution:



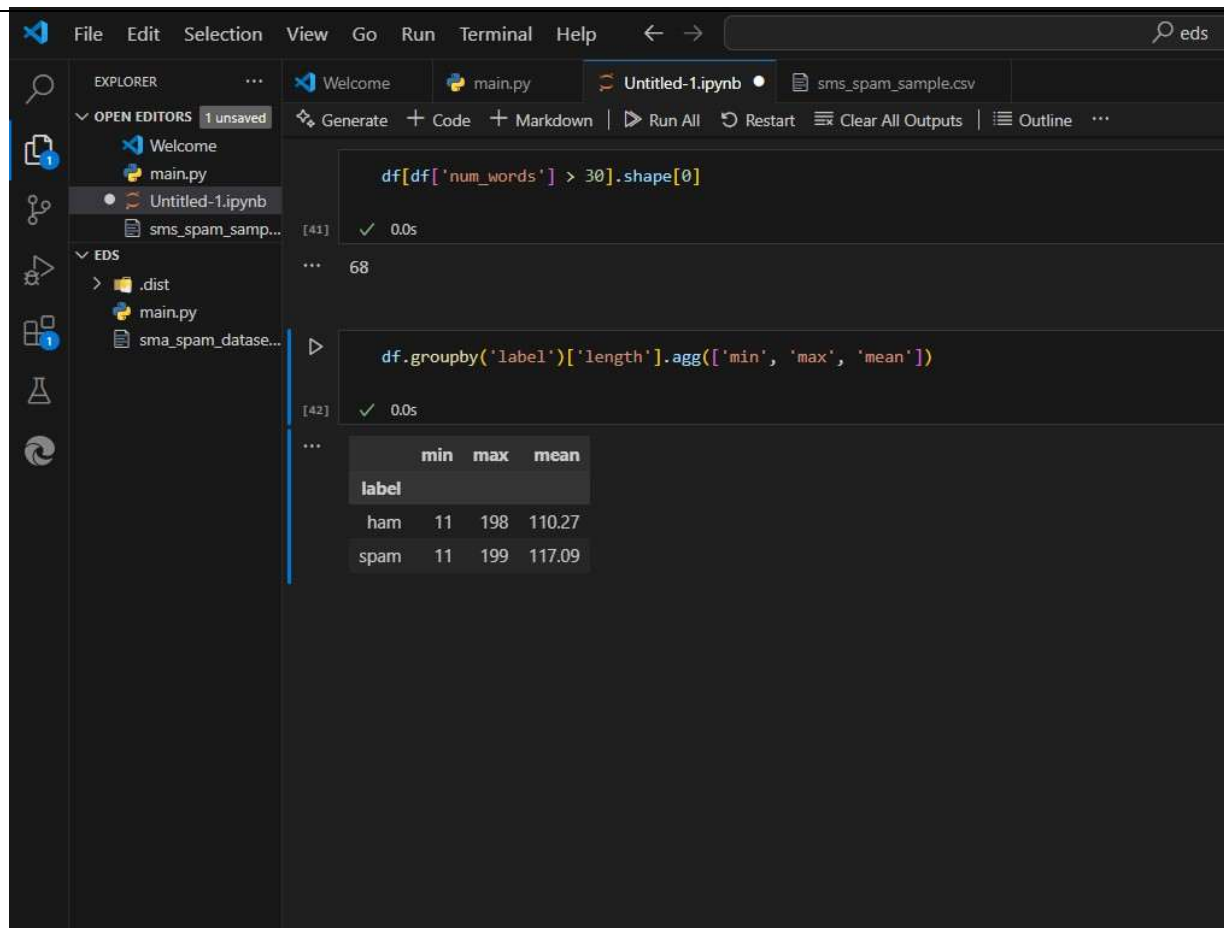
The screenshot shows a Jupyter Notebook interface in VS Code. The Explorer panel on the left shows the file structure with 'main.py' and 'sms_spam_sample.csv'. The main editor area displays a DataFrame with sample messages and their word counts. The code cell shows the following code:

```
df[df['num_words'] > 30].shape[0]
```

The output of the code is 68, indicating that there are 68 messages with more than 30 words.

10. Group by label and get max, min, mean of length

solution:



---end---