

EDS Activity 1

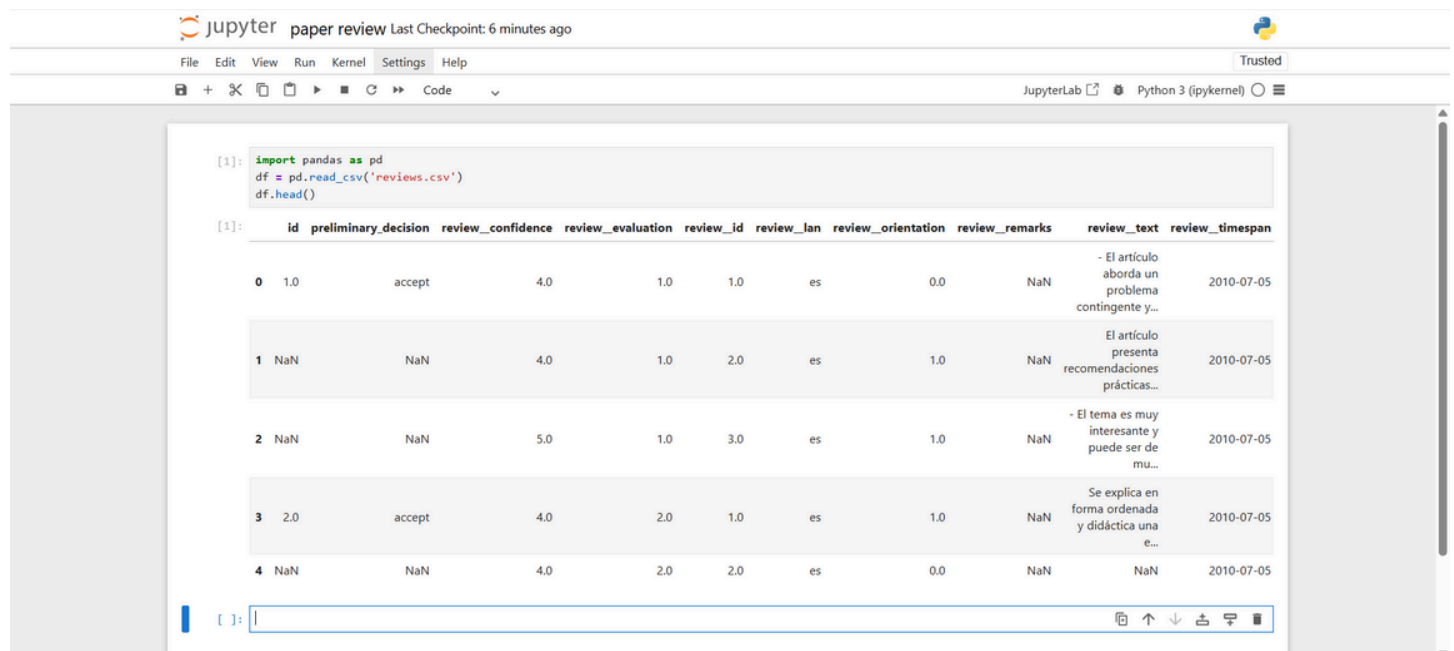
Name : Vedant Madankar

PRN : 202401040291

Roll No. : CS3-20

Dataset : Paper Review

Importing dataset into Jupyter Python Notebook



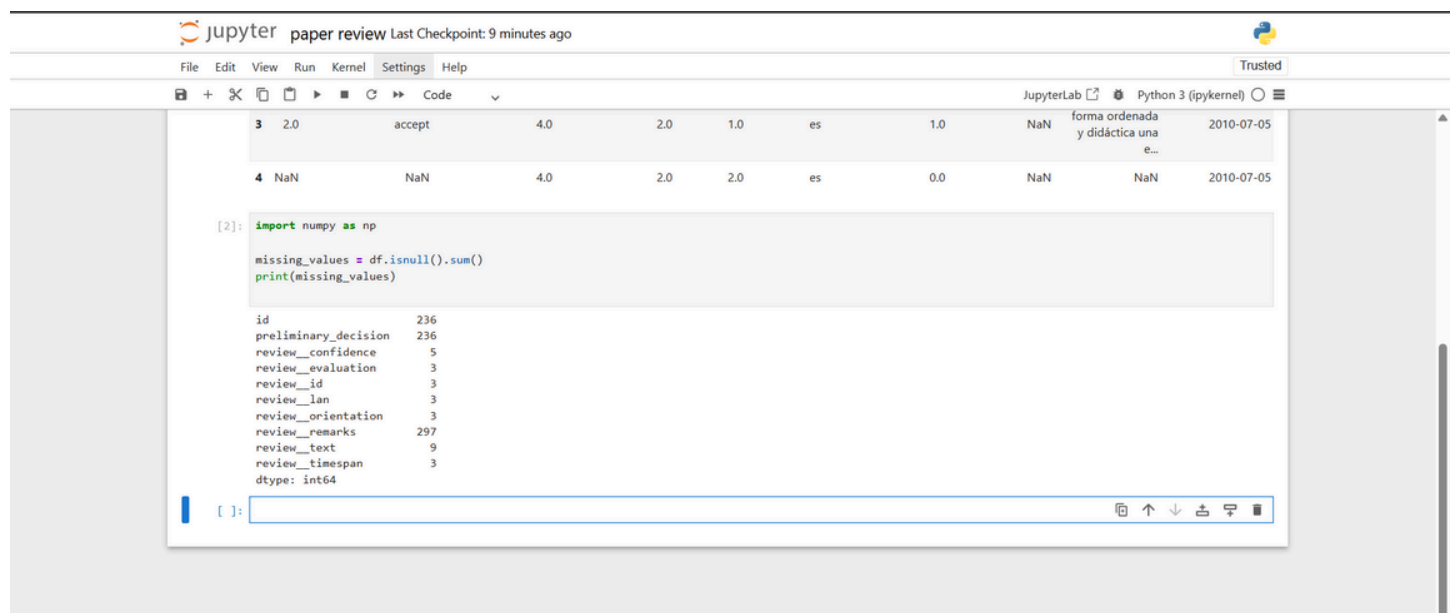
The screenshot shows a Jupyter Notebook interface with the title "paper review" and a last checkpoint of 6 minutes ago. The code cell [1] contains the following Python code:

```
[1]: import pandas as pd
df = pd.read_csv('reviews.csv')
df.head()
```

The output of the code is a pandas DataFrame with the following columns: id, preliminary_decision, review_confidence, review_evaluation, review_id, review_lang, review_orientation, review_remarks, review_text, and review_timespan. The first five rows of the DataFrame are displayed:

	id	preliminary_decision	review_confidence	review_evaluation	review_id	review_lang	review_orientation	review_remarks	review_text	review_timespan
0	1.0	accept	4.0	1.0	1.0	es	0.0	NaN	- El artículo aborda un problema contingente y...	2010-07-05
1	NaN	NaN	4.0	1.0	2.0	es	1.0	NaN	El artículo presenta recomendaciones prácticas...	2010-07-05
2	NaN	NaN	5.0	1.0	3.0	es	1.0	NaN	- El tema es muy interesante y puede ser de mu...	2010-07-05
3	2.0	accept	4.0	2.0	1.0	es	1.0	NaN	Se explica en forma ordenada y didáctica una e...	2010-07-05
4	NaN	NaN	4.0	2.0	2.0	es	0.0	NaN	NaN	2010-07-05

1. Finding the number of missing values in each column.



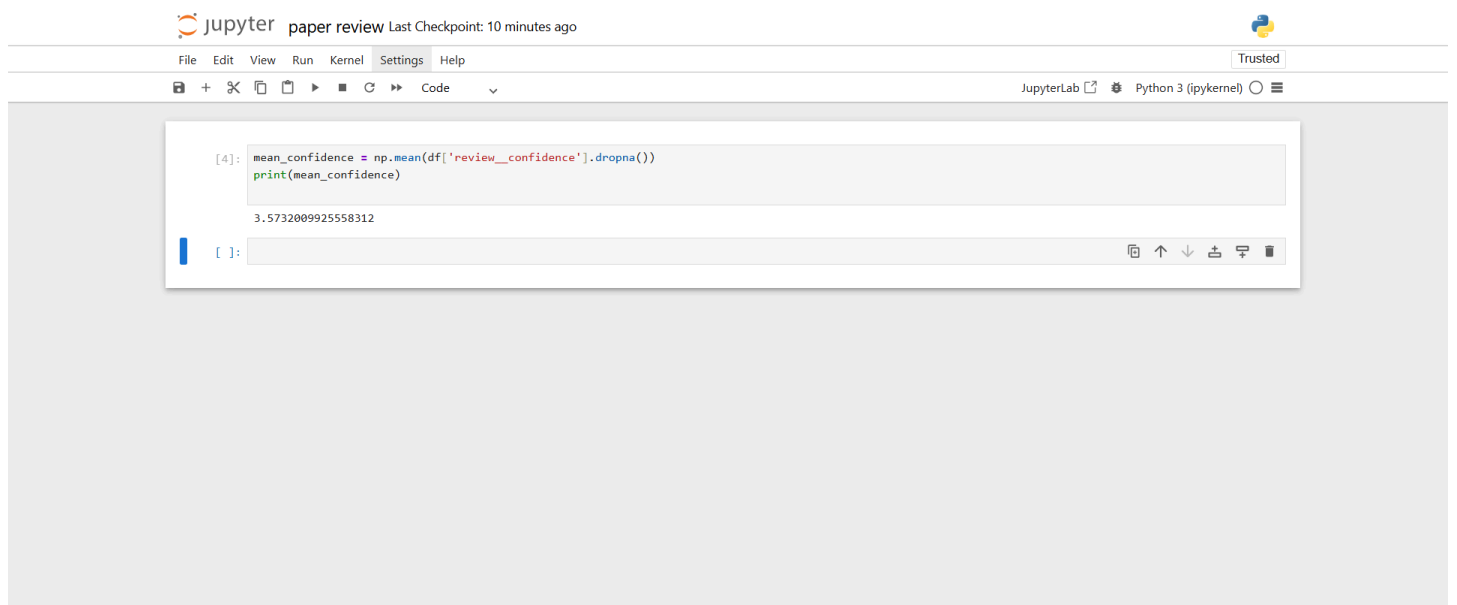
The screenshot shows a Jupyter Notebook interface with the title "paper review" and a last checkpoint of 9 minutes ago. The code cell [2] contains the following Python code:

```
[2]: import numpy as np
missing_values = df.isnull().sum()
print(missing_values)
```

The output of the code is a pandas Series showing the number of missing values in each column:

```
id                236
preliminary_decision  236
review_confidence      5
review_evaluation      3
review_id            3
review_lang          3
review_orientation    3
review_remarks       297
review_text          9
review_timespan       3
dtype: int64
```

2. Calculation of mean of the review confidence scores.



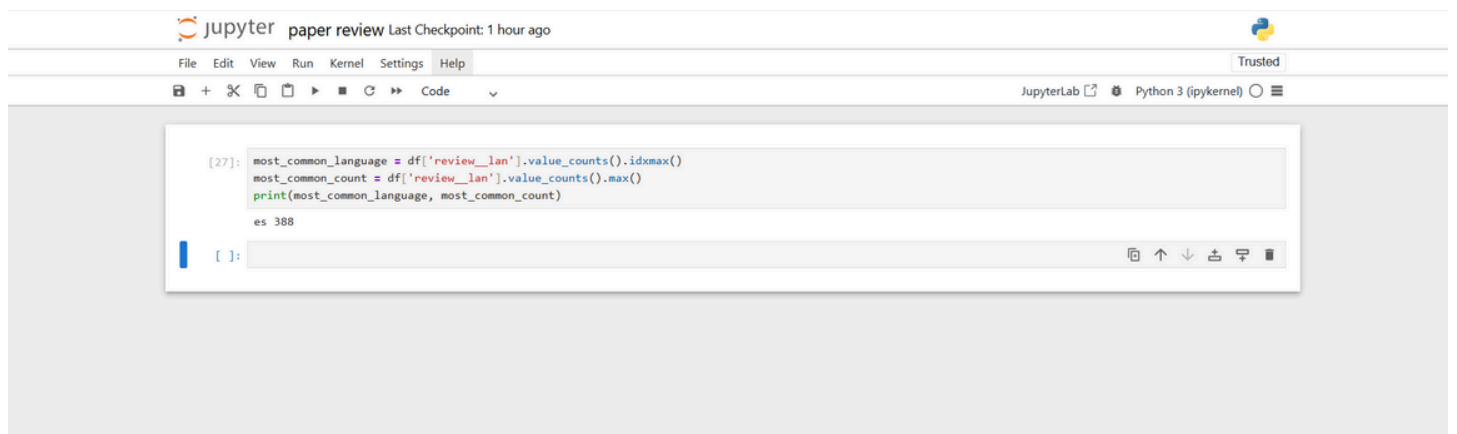
The JupyterLab interface shows a code cell with the following code:

```
[4]: mean_confidence = np.mean(df['review__confidence'].dropna())
print(mean_confidence)
```

The output of the code is:

```
3.5732009925558312
```

3.Find the most common review language and how many times it appears.



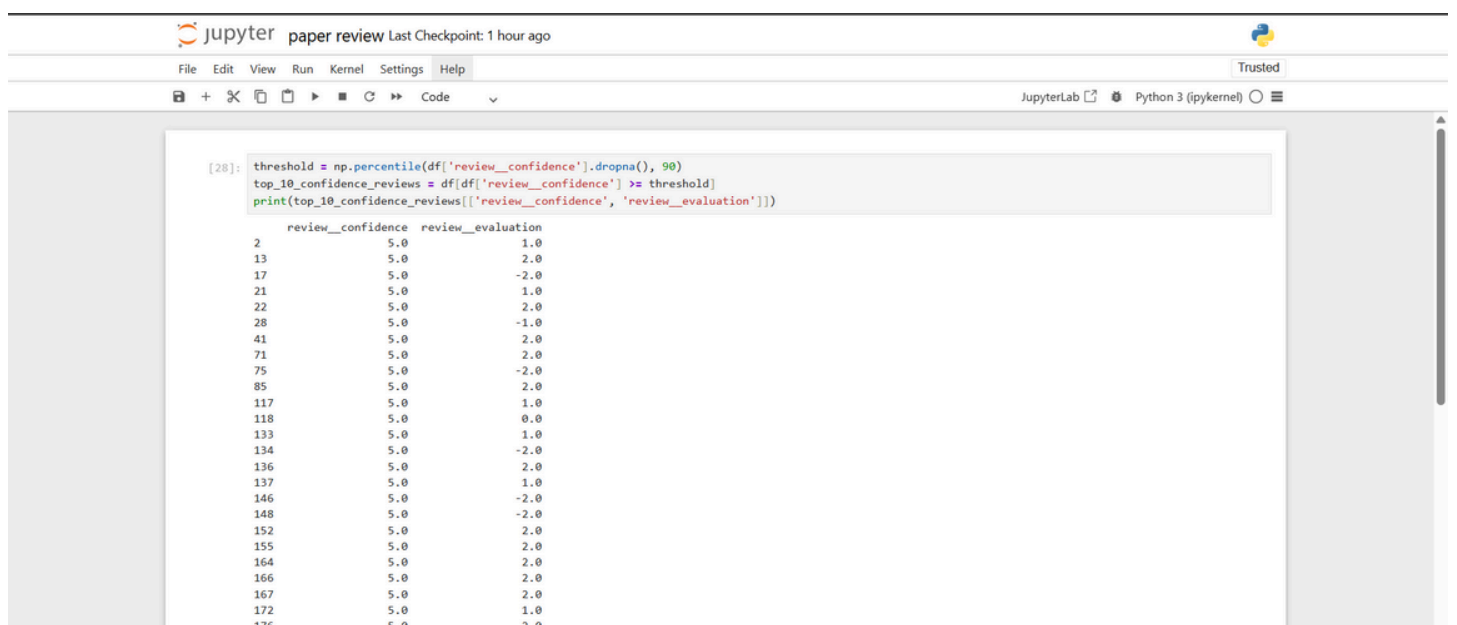
The JupyterLab interface shows a code cell with the following code:

```
[27]: most_common_language = df['review__lan'].value_counts().idxmax()
most_common_count = df['review__lan'].value_counts().max()
print(most_common_language, most_common_count)
```

The output of the code is:

```
es 388
```

4.Filter reviews where review confidence is in the top 10% of all values.



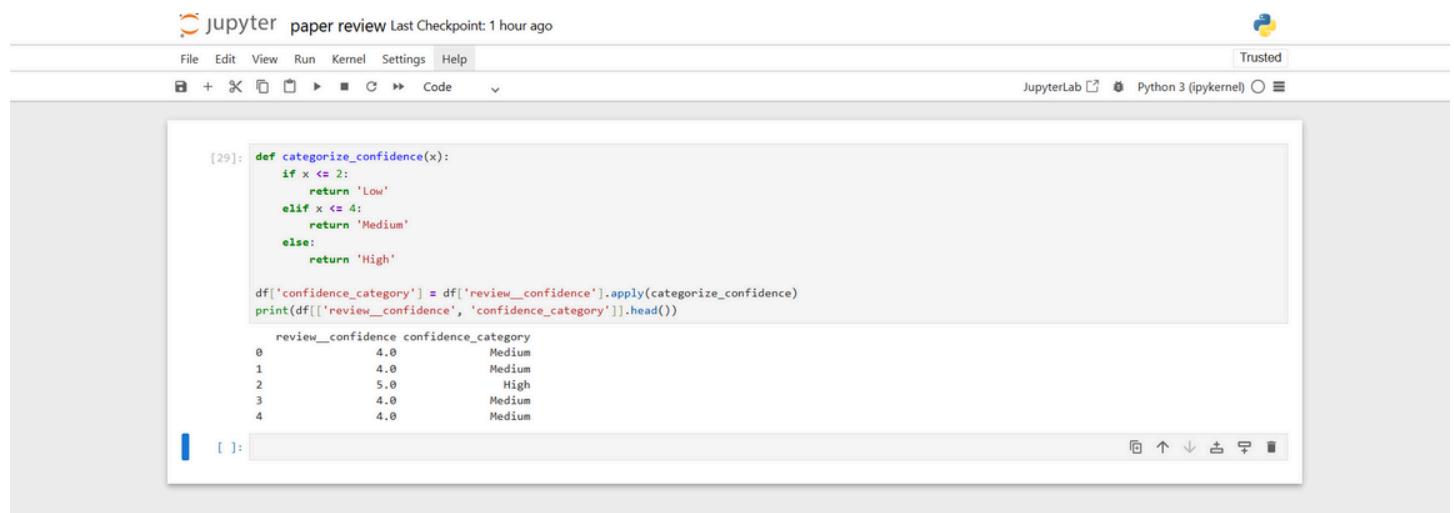
The JupyterLab interface shows a code cell with the following code:

```
[28]: threshold = np.percentile(df['review__confidence'].dropna(), 90)
top_10_confidence_reviews = df[df['review__confidence'] >= threshold]
print(top_10_confidence_reviews[['review__confidence', 'review_evaluation']])
```

The output of the code is a table with two columns: `review__confidence` and `review_evaluation`.

	review__confidence	review_evaluation
2	5.0	1.0
13	5.0	2.0
17	5.0	-2.0
21	5.0	1.0
22	5.0	2.0
28	5.0	-1.0
41	5.0	2.0
71	5.0	2.0
75	5.0	-2.0
85	5.0	2.0
117	5.0	1.0
118	5.0	0.0
133	5.0	1.0
134	5.0	-2.0
136	5.0	2.0
137	5.0	1.0
146	5.0	-2.0
148	5.0	-2.0
152	5.0	2.0
155	5.0	2.0
164	5.0	2.0
166	5.0	2.0
167	5.0	2.0
172	5.0	1.0
176	5.0	2.0

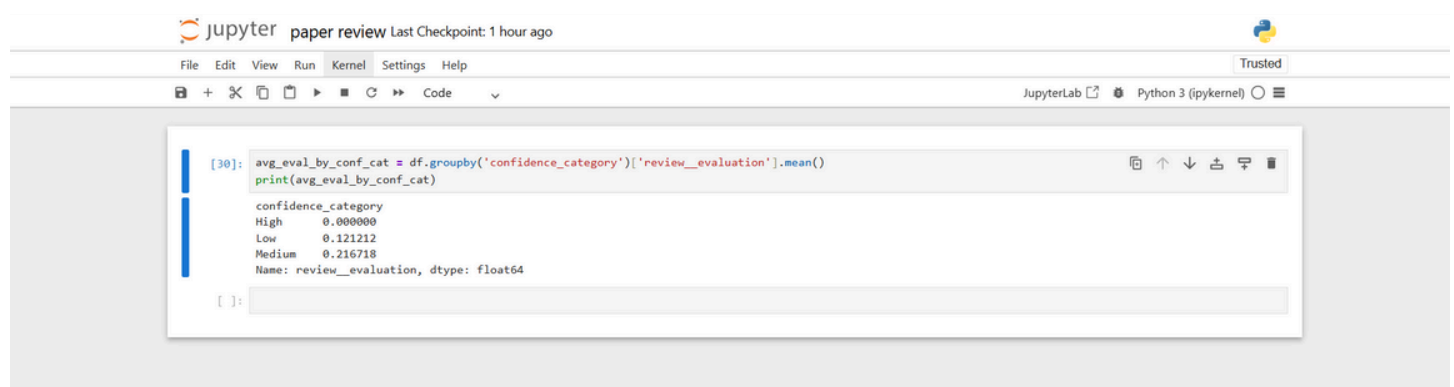
5. Create a new column confidence category:



The JupyterLab interface shows a code cell with the following Python code:

```
[29]: def categorize_confidence(x):  
    if x <= 2:  
        return 'Low'  
    elif x <= 4:  
        return 'Medium'  
    else:  
        return 'High'  
  
    df['confidence_category'] = df['review_confidence'].apply(categorize_confidence)  
    print(df[['review_confidence', 'confidence_category']].head())  
  
    review_confidence confidence_category  
0          4.0          Medium  
1          4.0          Medium  
2          5.0           High  
3          4.0          Medium  
4          4.0          Medium
```

6. Find the average review evaluation for each confidence category.



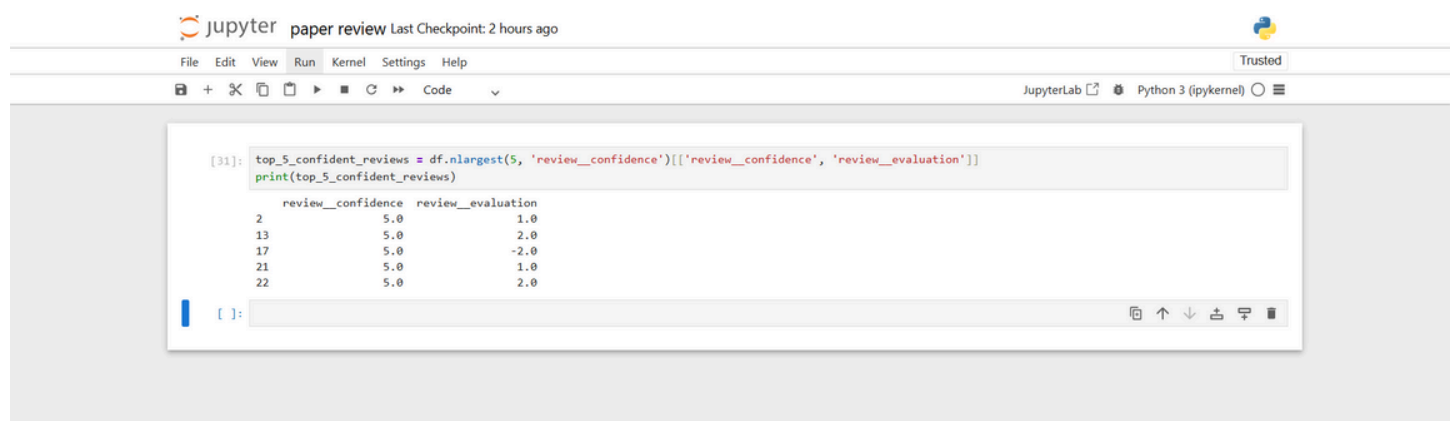
The JupyterLab interface shows a code cell with the following Python code:

```
[30]: avg_eval_by_conf_cat = df.groupby('confidence_category')['review_evaluation'].mean()  
print(avg_eval_by_conf_cat)
```

The output is a Series with the following values:

```
confidence_category  
High      0.000000  
Low       0.121212  
Medium    0.216718  
Name: review_evaluation, dtype: float64
```

7. Find the 5 reviews with the highest confidence and their evaluation scores.



The JupyterLab interface shows a code cell with the following Python code:

```
[31]: top_5_confident_reviews = df.nlargest(5, 'review_confidence')[['review_confidence', 'review_evaluation']]  
print(top_5_confident_reviews)
```

The output is a DataFrame with the following data:

	review_confidence	review_evaluation
2	5.0	1.0
13	5.0	2.0
17	5.0	-2.0
21	5.0	1.0
22	5.0	2.0

8. Calculate how many preliminary decisions are missing and what percentage that represents.



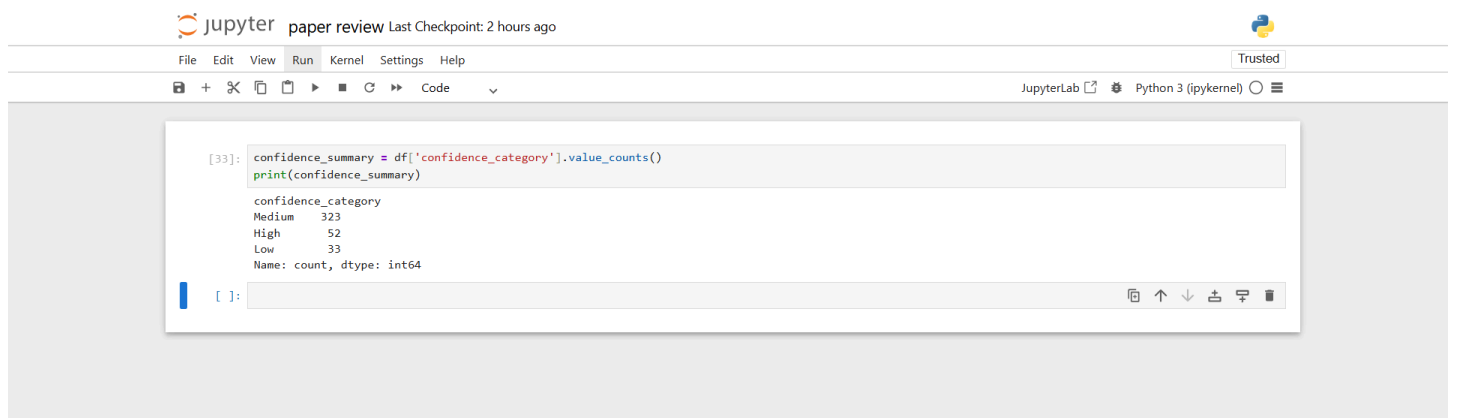
JupyterLab interface showing a code cell with the following Python code:

```
[32]: missing_decisions = df['preliminary_decision'].isnull().sum()
percentage_missing_decisions = (missing_decisions / len(df)) * 100
print(f"Missing decisions: {missing_decisions} ({percentage_missing_decisions:.2f}%)")
```

The output of the code is:

```
Missing decisions: 236 (57.84%)
```

9.Create a summary dataframe with counts of 'Low', 'Medium', and 'High' confidence categories.



JupyterLab interface showing a code cell with the following Python code:

```
[33]: confidence_summary = df['confidence_category'].value_counts()
print(confidence_summary)
```

The output of the code is:

```
confidence_category
Medium    323
High       52
Low        33
Name: count, dtype: int64
```

10.Identify columns where more than 30% of the data is missing. List those columns.



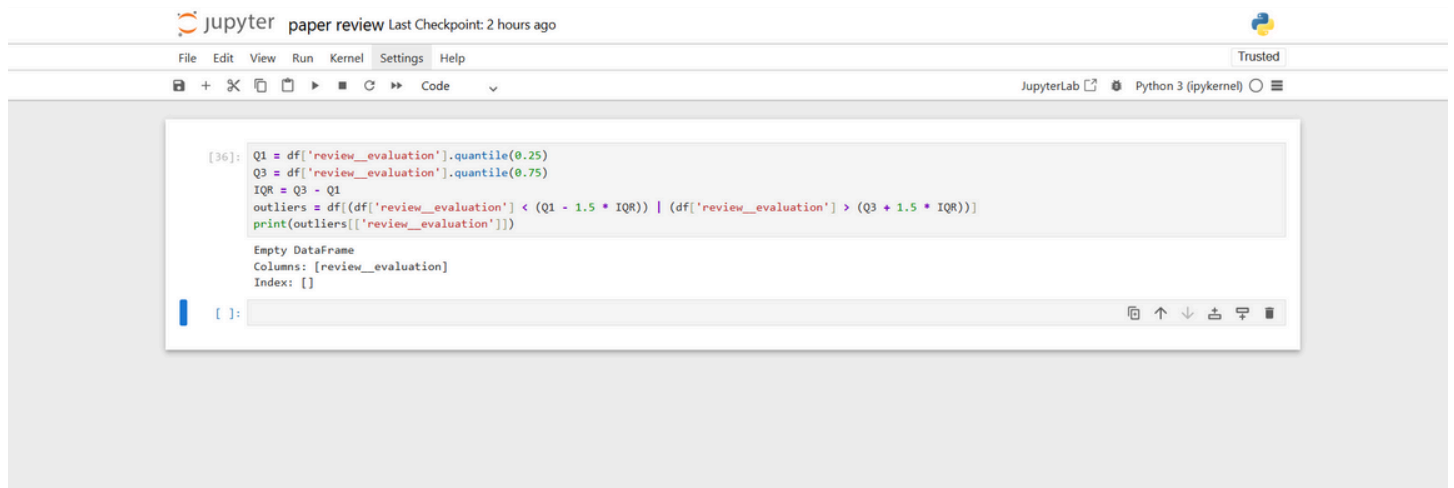
JupyterLab interface showing a code cell with the following Python code:

```
[35]: missing_percentage = df.isnull().mean() * 100
columns_with_high_missing = missing_percentage[missing_percentage > 30].index.tolist()
print(columns_with_high_missing)
```

The output of the code is:

```
['id', 'preliminary_decision', 'review_remarks']
```

11.Calculate the interquartile range (IQR) of review evaluation and identify outlier reviews.

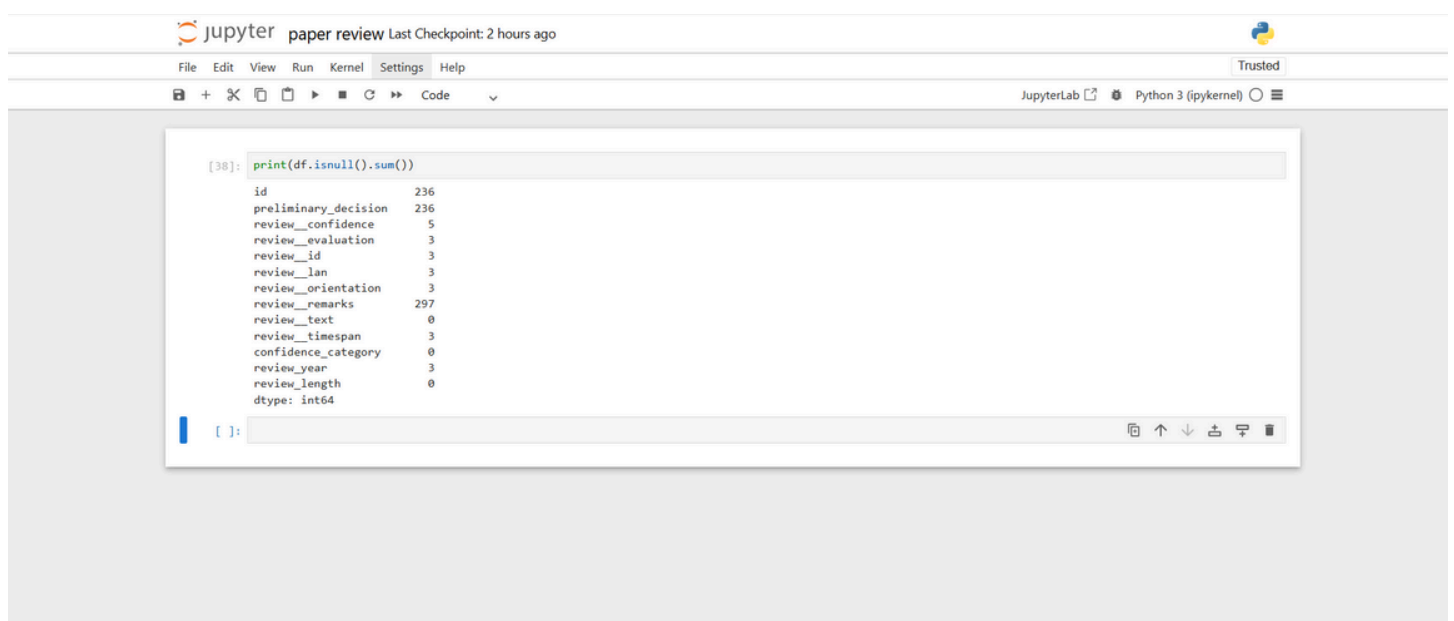


The screenshot shows a JupyterLab notebook titled 'paper review' with a 'Trusted' security level. The code cell [36] contains the following Python code:

```
[36]: Q1 = df['review_evaluation'].quantile(0.25)
      Q3 = df['review_evaluation'].quantile(0.75)
      IQR = Q3 - Q1
      outliers = df[(df['review_evaluation'] < (Q1 - 1.5 * IQR)) | (df['review_evaluation'] > (Q3 + 1.5 * IQR))]
      print(outliers[['review_evaluation']])
```

The output of the code is an 'Empty DataFrame' with columns: ['review_evaluation'] and index: [].

12. Find the number of missing values in each column.



The screenshot shows a JupyterLab notebook titled 'paper review' with a 'Trusted' security level. The code cell [38] contains the following Python code:

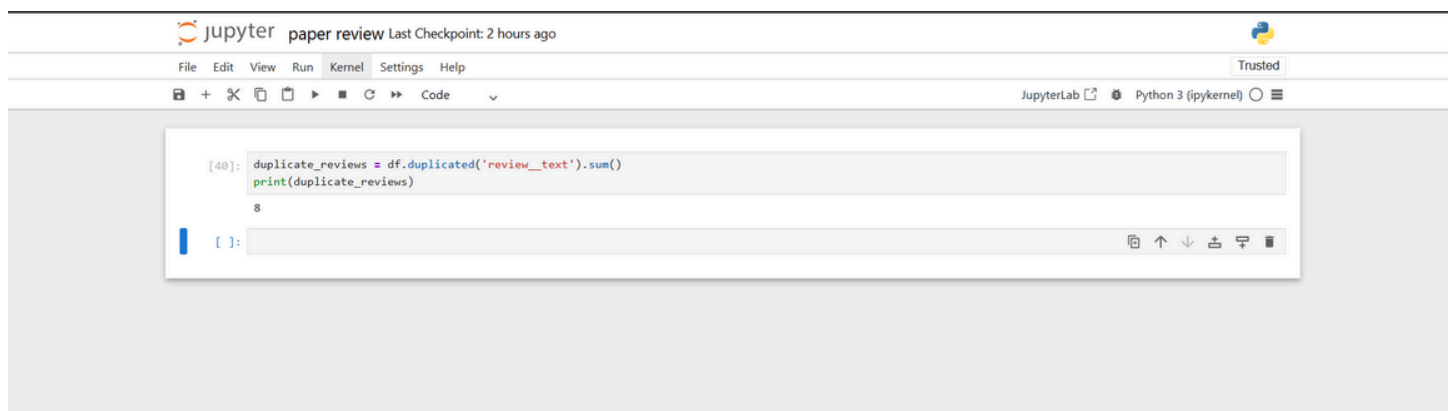
```
[38]: print(df.isnull().sum())
```

The output of the code is a series of missing value counts for each column:

Column	Count
id	236
preliminary_decision	236
review_confidence	5
review_evaluation	3
review_id	3
review_lang	3
review_orientation	3
review_remarks	297
review_text	0
review_timespan	3
confidence_category	0
review_year	3
review_length	0

The dtype for the output is int64.

13. Check if any duplicate review texts exist.



The screenshot shows a JupyterLab notebook titled 'paper review' with a 'Trusted' security level. The code cell [40] contains the following Python code:

```
[40]: duplicate_reviews = df.duplicated('review_text').sum()
      print(duplicate_reviews)
```

The output of the code is the number of duplicate review texts, which is 8.

14. Find the average length of review texts.

```
[41]: df['review_length'] = df['review_text'].apply(lambda x: len(x))
average_length = df['review_length'].mean()
print(average_length)
998.0
```

15. Find the number of reviews written in the year 2015.

```
[42]: reviews_2015 = df[df['review_year'] == 2015]
print(len(reviews_2015))
80
```

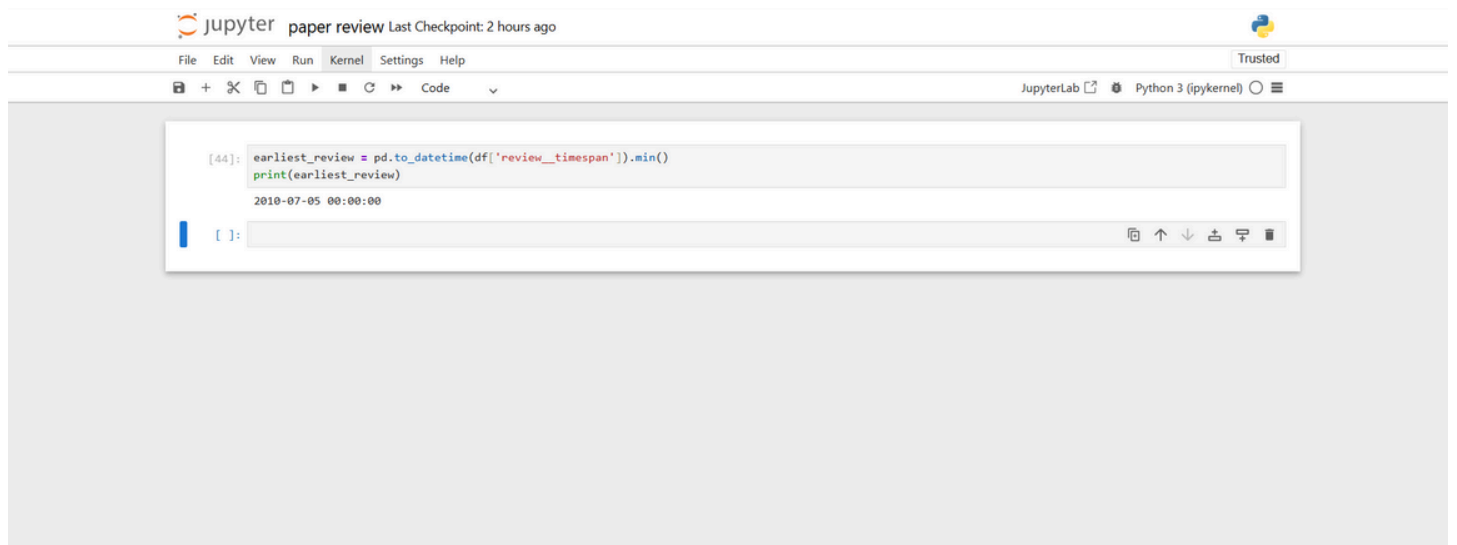
16. Get a summary of numeric columns using describe().

```
[43]: print(df.describe())
```

	id	review__confidence	review_evaluation	review_id \
count	172.000000	403.000000	405.000000	405.000000
mean	86.500000	3.573201	0.182716	1.824691
std	49.796252	0.844341	1.502868	0.821362
min	1.000000	1.000000	-2.000000	1.000000
25%	43.750000	3.000000	-1.000000	1.000000
50%	86.500000	4.000000	0.000000	2.000000
75%	129.250000	4.000000	2.000000	2.000000
max	172.000000	5.000000	2.000000	4.000000

	review__orientation	review_year	review_length
count	405.000000	405.000000	408.000000
mean	-0.212346	2012.839506	998.000000
std	1.019292	1.919524	832.942786
min	-2.000000	2010.000000	18.000000
25%	-1.000000	2010.000000	462.750000
50%	0.000000	2014.000000	793.000000
75%	1.000000	2014.000000	1269.750000
max	2.000000	2015.000000	6345.000000

17. Find the earliest (oldest) review date.

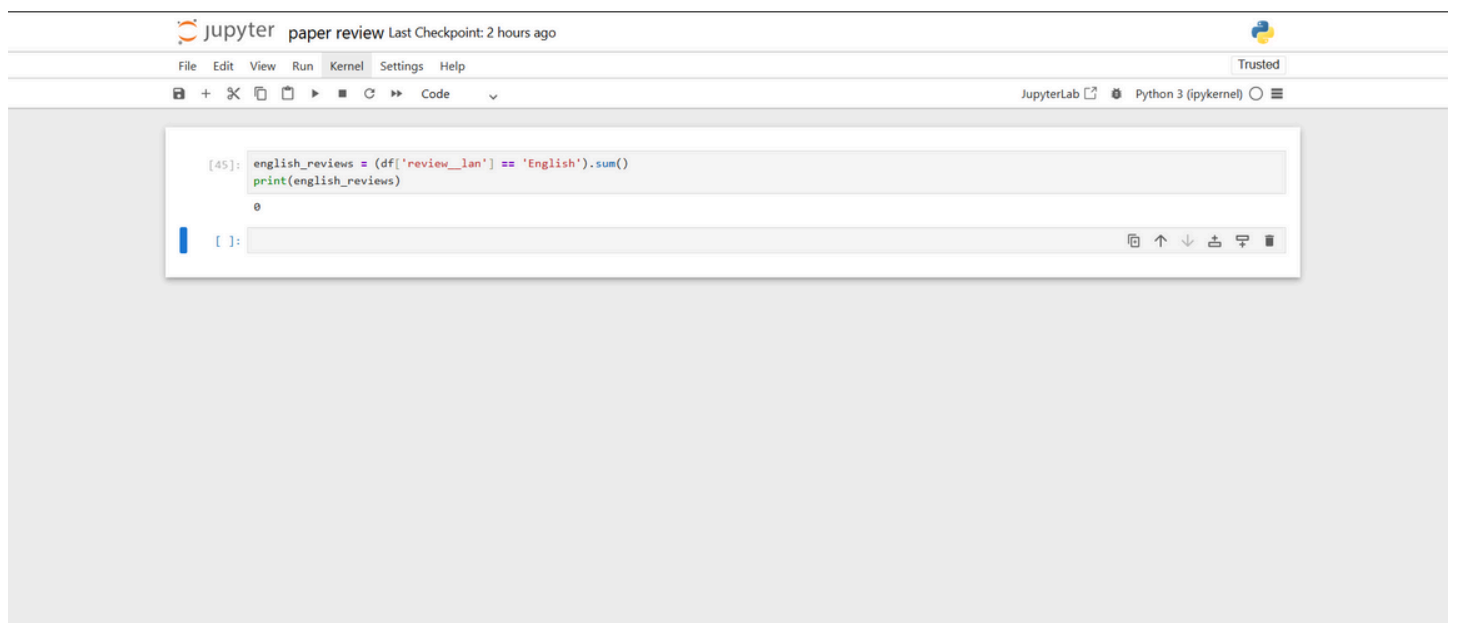


The screenshot shows a JupyterLab window titled "paper review" with a last checkpoint 2 hours ago. The interface includes a top bar with "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help" menus. Below the menu bar is a toolbar with icons for file operations and a "Code" button. The main area contains a code cell with the following Python code:

```
[44]: earliest_review = pd.to_datetime(df['review_timespan']).min()
      print(earliest_review)
      2010-07-05 00:00:00
```

The output of the code is displayed below the code cell.

18. Find how many reviews are written in English.

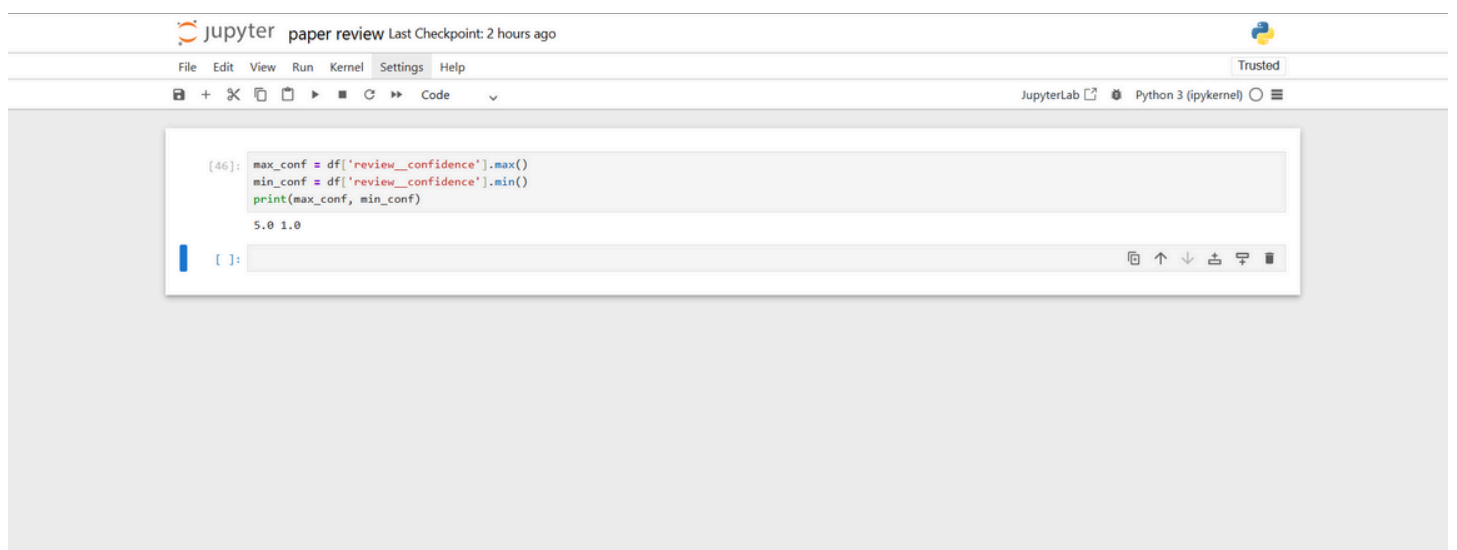


The screenshot shows a JupyterLab window titled "paper review" with a last checkpoint 2 hours ago. The interface includes a top bar with "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help" menus. Below the menu bar is a toolbar with icons for file operations and a "Code" button. The main area contains a code cell with the following Python code:

```
[45]: english_reviews = (df['review_lang'] == 'English').sum()
      print(english_reviews)
      0
```

The output of the code is displayed below the code cell.

19. Find the maximum and minimum review confidence values.

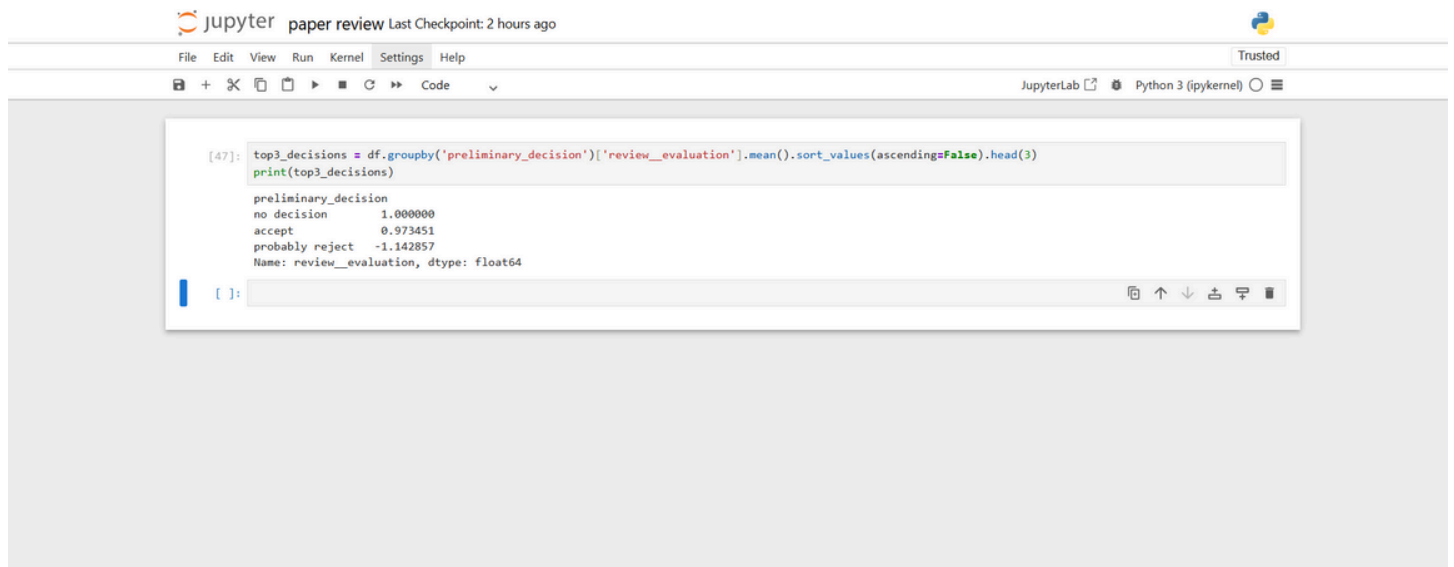


The screenshot shows a JupyterLab window titled "paper review" with a last checkpoint 2 hours ago. The interface includes a top bar with "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help" menus. Below the menu bar is a toolbar with icons for file operations and a "Code" button. The main area contains a code cell with the following Python code:

```
[46]: max_conf = df['review_confidence'].max()
      min_conf = df['review_confidence'].min()
      print(max_conf, min_conf)
      5.0 1.0
```

The output of the code is displayed below the code cell.

20. Find the top 3 preliminary decisions having the highest average review evaluation.



The image shows a JupyterLab interface with a code cell containing the following Python code:

```
[47]: top3_decisions = df.groupby('preliminary_decision')['review_evaluation'].mean().sort_values(ascending=False).head(3)
      print(top3_decisions)
```

The output of the code is a pandas Series:

```
preliminary_decision
no decision          1.000000
accept              0.973451
probably reject     -1.142857
Name: review_evaluation, dtype: float64
```

The JupyterLab interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help), a toolbar with icons for file operations, and a status bar at the bottom indicating the current kernel is Python 3 (ipykernel).