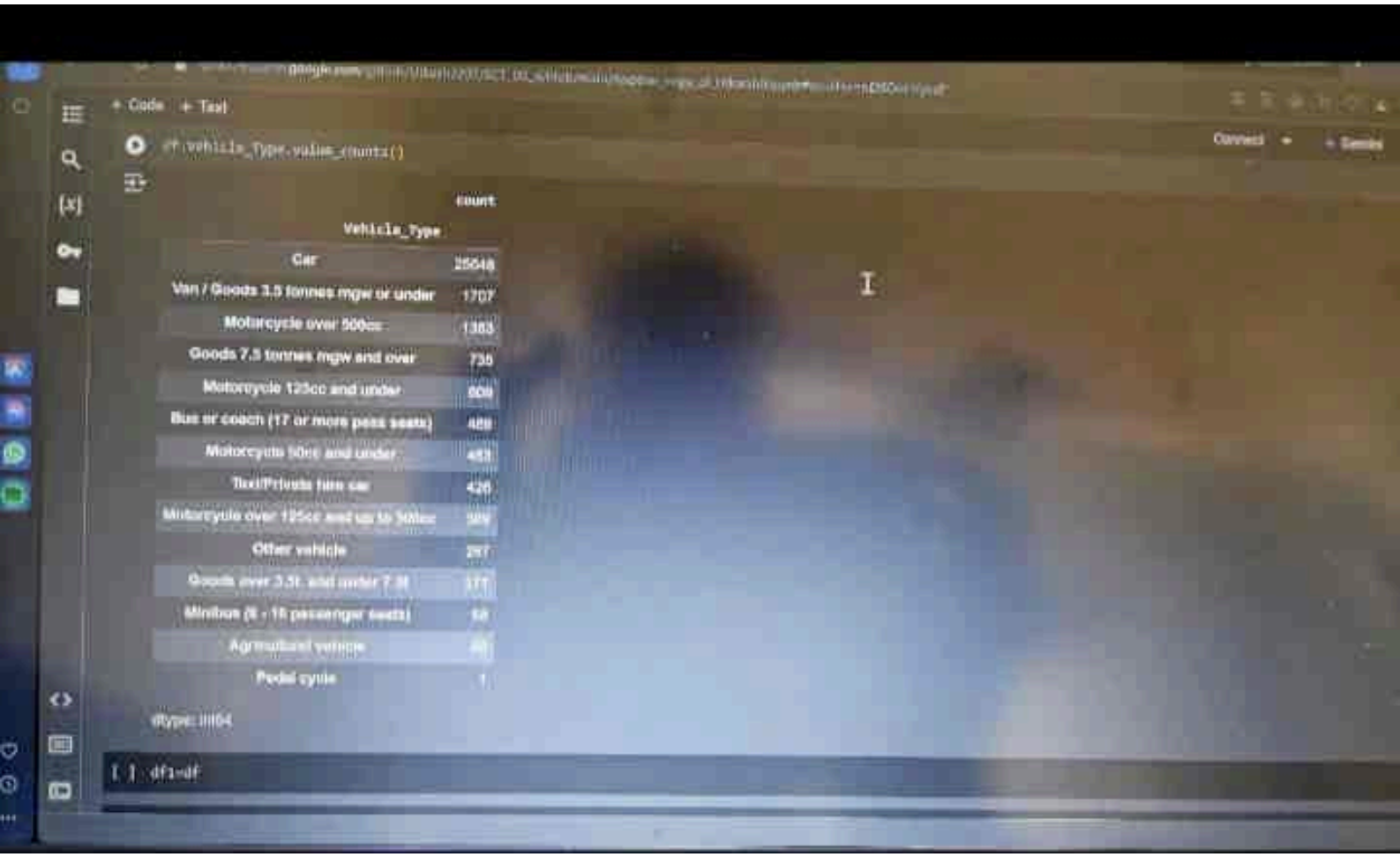
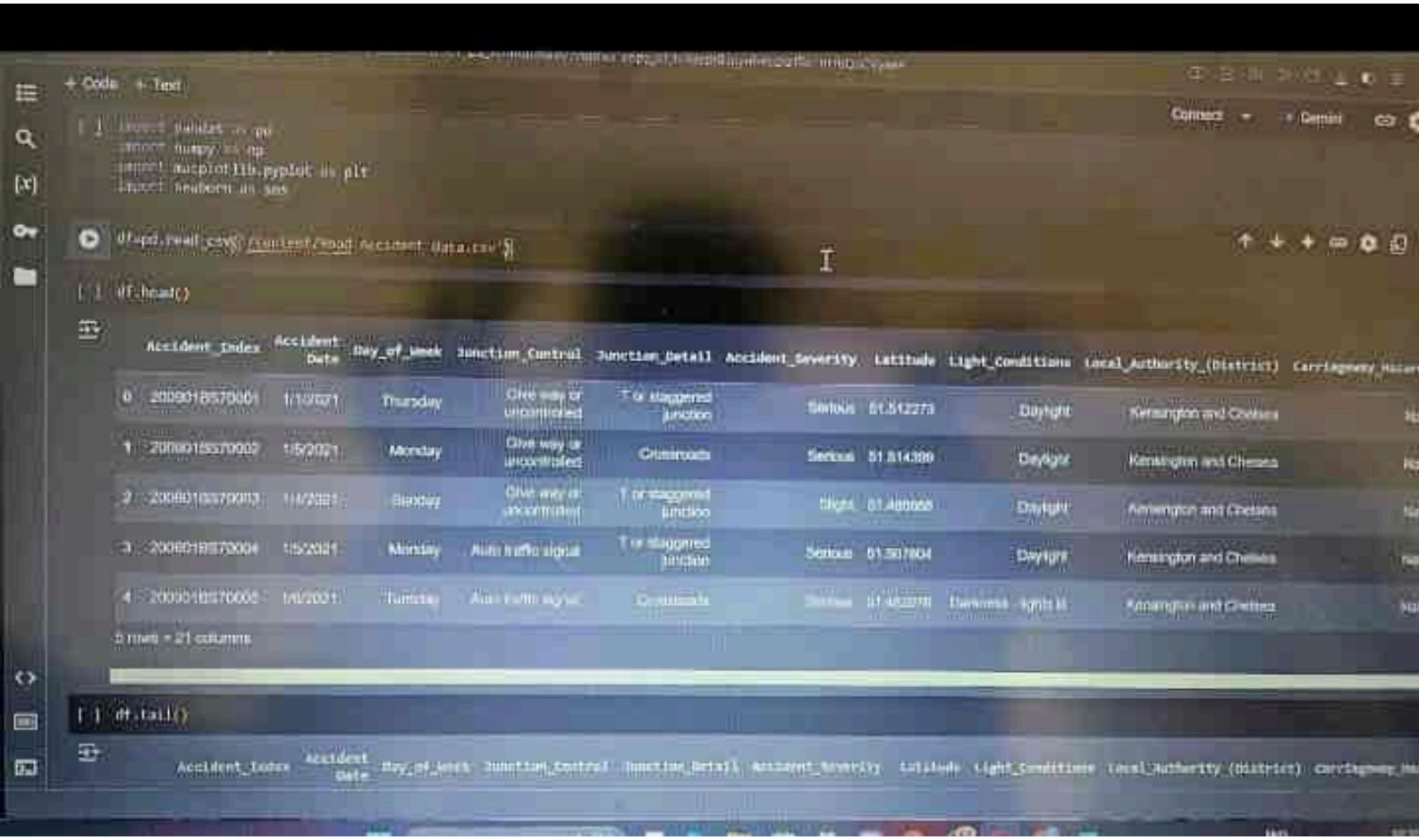

+ Code + Text
df.Vehicle_Type.value_counts()
count
Vehicle_Type
Car 25548
Van / Goods 3.5 tonnes mgw or under 1707
Motorcycle over 500cc 1383
Goods 7.5 tonnes mgw and over 735
Motorcycle 125cc and under 609
Bus or coach (17 or more pass seats) 469
Motorcycle 50cc and under 453
Taxi/Private hire car 426
Other vehicle 287
Minibus (8 - 16 passenger seats) 58
Pedal cycle 1
dtype: int64
df1=df


+ Code + Text
Connect
Gemini
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/Road Accident Data.csv')
df.head()
Accident_Index
Accident Date
Day_of_Week
Junction_Control
Junction_Detail
Accident_Severity
Latitude
Light_Conditions
Local_Authority_(District)
2009018570001
1/1/2021
Thursday
Give way or uncontrolled
T or staggered junction
Serious
51.512273
Daylight
Kensington and Chelsea
2009018570002
1/5/2021
Monday
Give way or uncontrolled
Crossroads
Serious
51.514389
Daylight
Kensington and Chelsea
2009018570003
1/6/2021
Monday
Give way or uncontrolled
T or staggered junction
Slight
51.488868
Daylight
Kensington and Chelsea
2009018570004
1/5/2021
Monday
Auto traffic signal
T or staggered junction
Serious
51.507804
Daylight
Kensington and Chelsea
2009018570005
1/6/2021
Tuesday
Auto traffic signal
Crossroads
Serious
51.482778
Darkness - lights lit
Kensington and Chelsea
5 rows × 21 columns
df.tail()
Accident_Index
Accident Date
Day_of_Week
Junction_Control
Junction_Detail
Accident_Severity
Latitude
Light_Conditions
Local_Authority_(District)

+ Code + Text

Connect + Gemini

```
df.shape
```

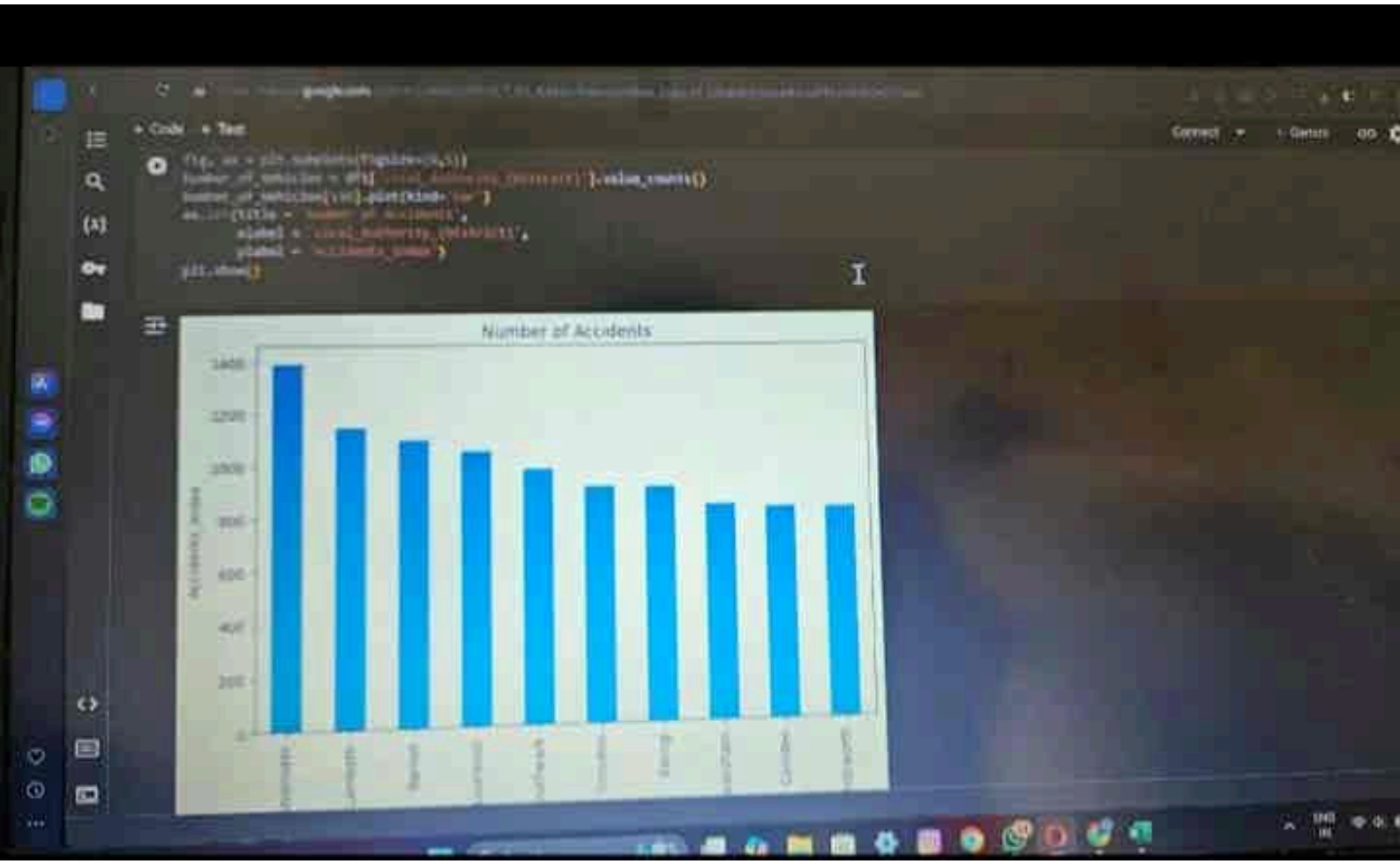
```
(32432, 21)
```

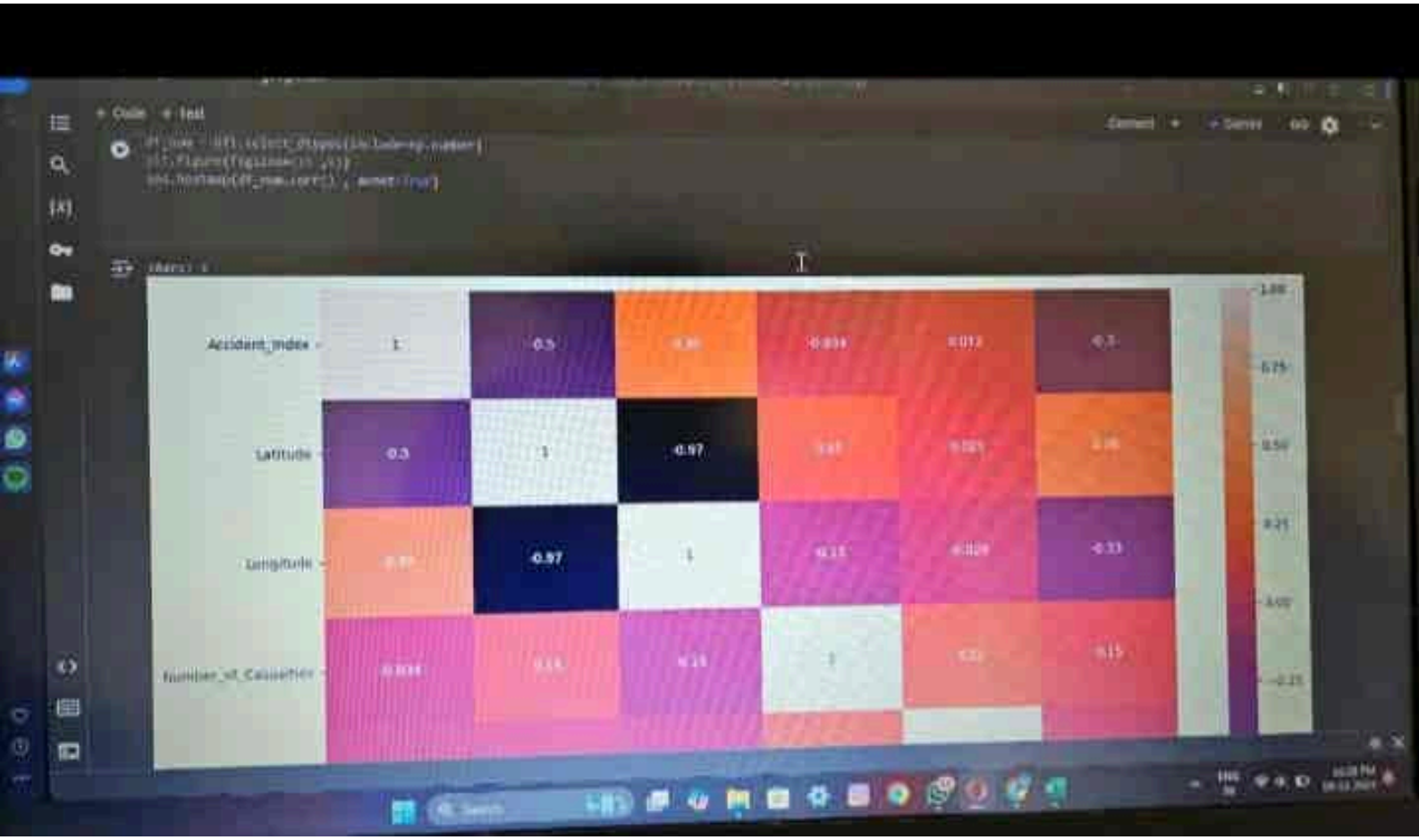
```
df.describe()
```

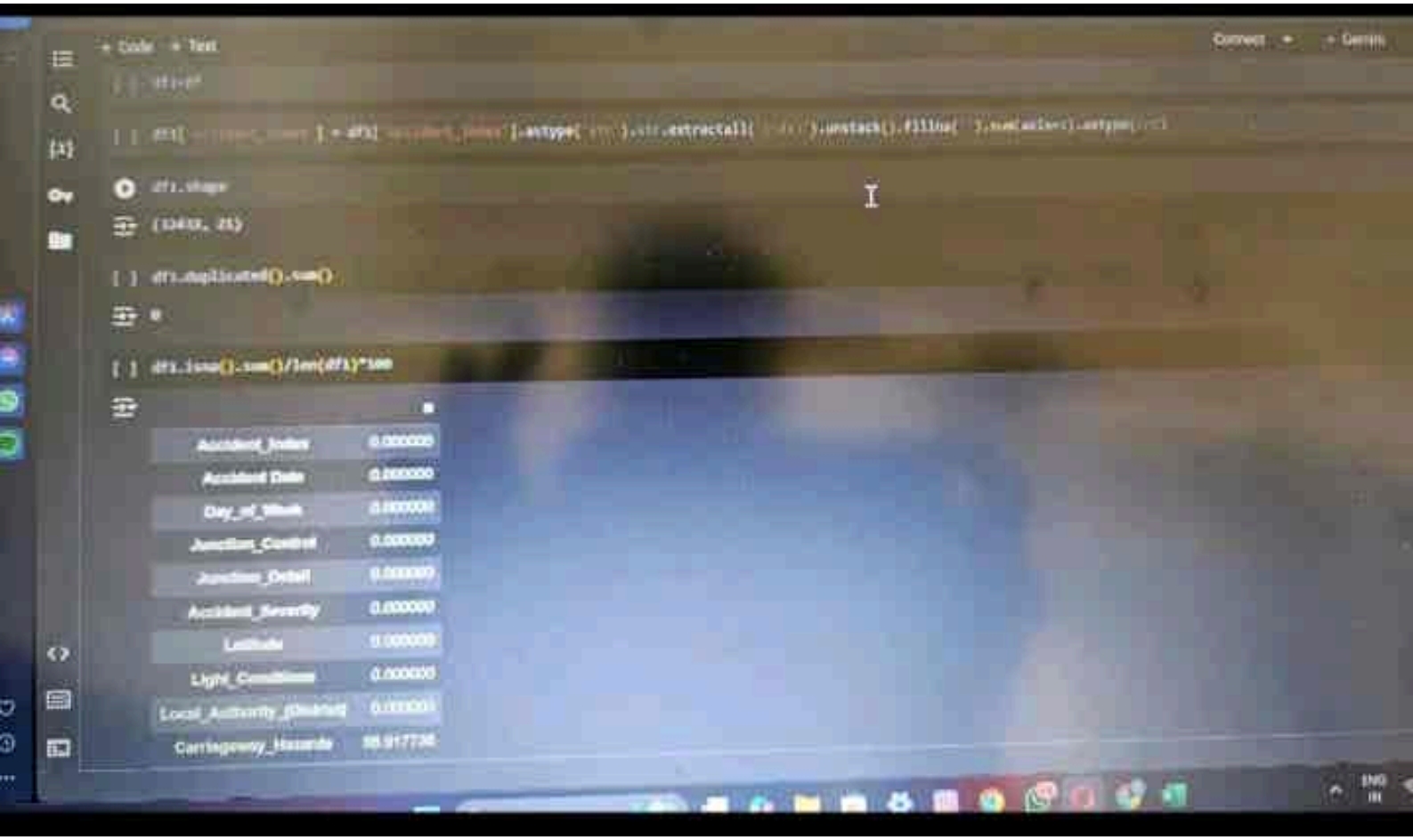
	Latitude	Longitude	Number_of_Casualties	Number_of_Vehicles	Speed_limit
count	32432.000000	32432.000000	32432.000000	32432.000000	32432.000000
mean	52.154306	-0.888065	1.271337	1.780829	32.884800
std	1.053300	1.230761	0.686682	0.634791	8.659488
min	51.280164	-3.097103	1.000000	1.000000	20.000000
25%	51.409689	-2.440578	1.000000	1.000000	30.000000
50%	51.544813	-0.184500	1.000000	2.000000	30.000000
75%	53.406021	0.073284	1.000000	2.000000	30.000000
max	56.128572	0.297740	13.000000	6.000000	70.000000

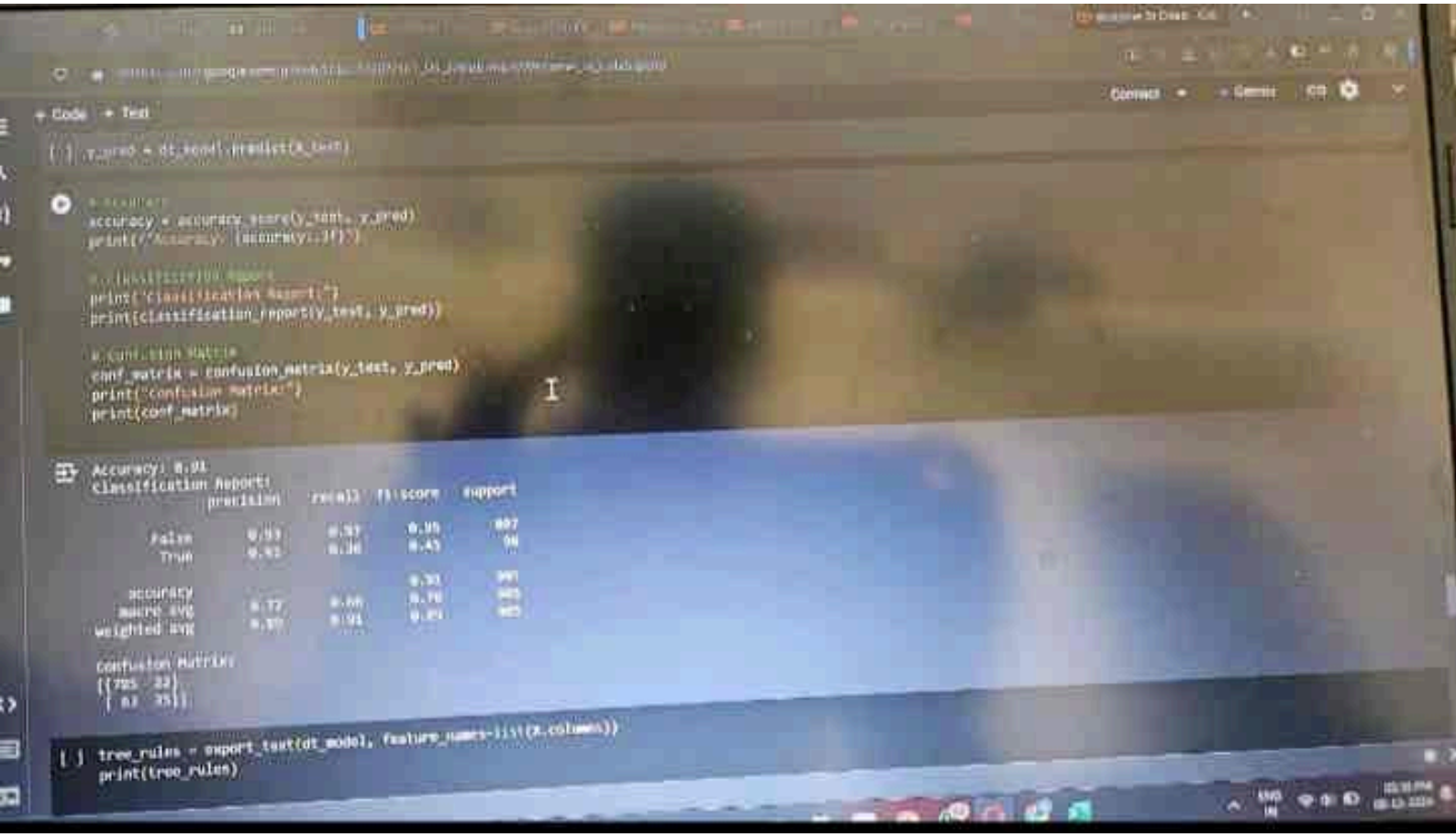
```
df.Vehicle_Type.value_counts()
```

Vehicle_Type	count
Car	25048
Van / Goods 3.5 tonnes mgw or under	1707
Motorcycle over 500cc	1307
Goods 7.5 tonnes mgw and over	730


+ Code
+ Text
Connect
Number of Accidents


Accident_Index
Latitude
Longitude
Number_of_Casualties
1
0.3
0.97
0.97
1
0.3
0.97
1


+ Code + Text
Connect
df1.shape
(1433, 21)
df1.duplicated().sum()
0
df1.isna().sum()/len(df1)*100
Accident_Index 0.000000
Accident Date 0.000000
Day_of_Week 0.000000
Junction_Control 0.000000
Junction_Detail 0.000000
Accident_Severity 0.000000
Latitude 0.000000
Light_Conditions 0.000000
Local_Authority_(District) 0.000000
Carriageway_Hazards 99.917736


+ Code + Text
y_pred = dt_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
Accuracy: 0.91
Classification Report:
precision recall f1-score support
False 0.93 0.97 0.95 807
True 0.83 0.68 0.75 193
accuracy 0.91 1000
macro avg 0.88 0.82 0.85 1000
weighted avg 0.91 0.91 0.91 1000
Confusion Matrix:
[[785 22]
[62 131]]
tree_rules = export_text(dt_model, feature_names=list(X.columns))
print(tree_rules)