Upload Dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose files crime_vs_s…c_factors.csv

- **crime_vs_socioeconomic_factors.csv**(text/csv) - 80782 bytes, last modified: 09/05/2025 - 100% done

Saving crime_vs_socioeconomic_factors.csv to crime_vs_socioeconomic_factors (1).csv

Load Dataset

```
import pandas as pd

# Load dataset
df = pd.read_csv('crime_vs_socioeconomic_factors.csv')
df.head()
```

	Region	Crime_Rate	Education_Level	Employment_Rate	Median_Income	Poverty_Rate	Population_Density
0	Region_1	1176	76.492001	73.315344	116664	14.655300	1523
1	Region_2	910	85.361505	46.088096	21401	19.712623	3293
2	Region_3	1344	88.388975	63.911701	105179	9.634319	4528
3	Region_4	1180	54.364509	65.305378	35193	19.994283	2231
4	Region_5	1145	75.305198	51.627568	112389	26.537843	1499

Next steps: Generate code with df | View recommended plots | New interactive sheet

Data Exploration

```
# Dataset shape
print("Shape of the dataset:", df.shape)
```

```
Shape of the dataset: (1000, 7)
```

```
# Columns and data types
print("Columns:", df.columns.tolist())
df.info()
```

```
Columns: ['Region', 'Crime_Rate', 'Education_Level', 'Employment_Rate', 'Median_Income', 'Poverty_Rate', 'Population_Density']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Region              1000 non-null   object
 1   Crime_Rate          1000 non-null   int64
 2   Education_Level     1000 non-null   float64
 3   Employment_Rate     1000 non-null   float64
 4   Median_Income       1000 non-null   int64
 5   Poverty_Rate        1000 non-null   float64
 6   Population_Density  1000 non-null   int64
dtypes: float64(3), int64(3), object(1)
memory usage: 54.8+ KB
```

```
# Descriptive statistics
df.describe()
```

	Crime_Rate	Education_Level	Employment_Rate	Median_Income	Poverty_Rate	Population_Density
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	802.428000	75.391052	64.820880	69427.926000	17.465860	2516.161000
std	418.186261	14.120708	14.652422	29219.031581	7.226494	1414.815908
min	50.000000	50.011876	40.009420	20112.000000	5.004663	51.000000
25%	452.750000	63.229627	51.631100	44347.000000	11.237297	1277.750000
50%	818.500000	76.140098	65.034601	67484.000000	17.550832	2606.000000
75%	1159.000000	87.409706	77.481090	96258.250000	23.859603	3677.500000
max	1495.000000	99.967675	89.985690	119977.000000	29.940571	4993.000000

Check for missing values and duplicates

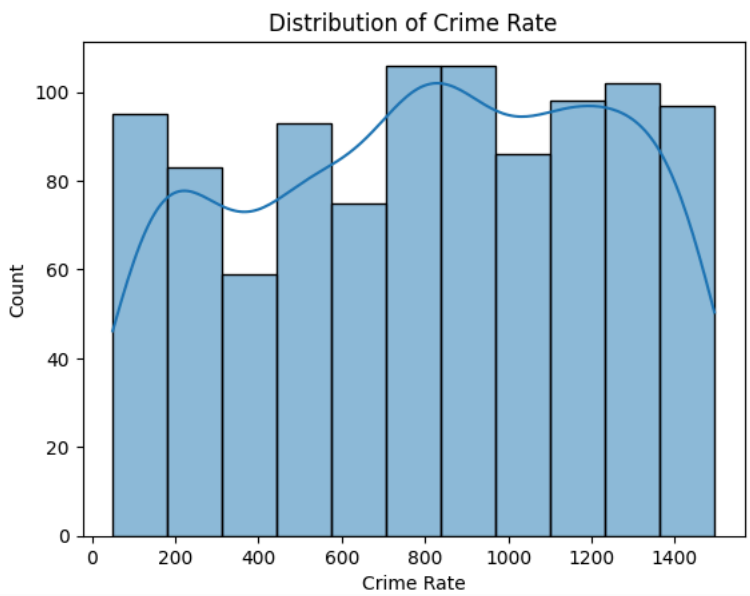
```
# Check for missing values and duplicates
print("Missing Values:\n", df.isnull().sum())
print("Duplicate Rows:", df.duplicated().sum())
```

```
Missing Values:
 Region                0
Crime_Rate            0
Education_Level       0
Employment_Rate       0
Median_Income         0
Poverty_Rate          0
Population_Density    0
dtype: int64
Duplicate Rows: 0
```

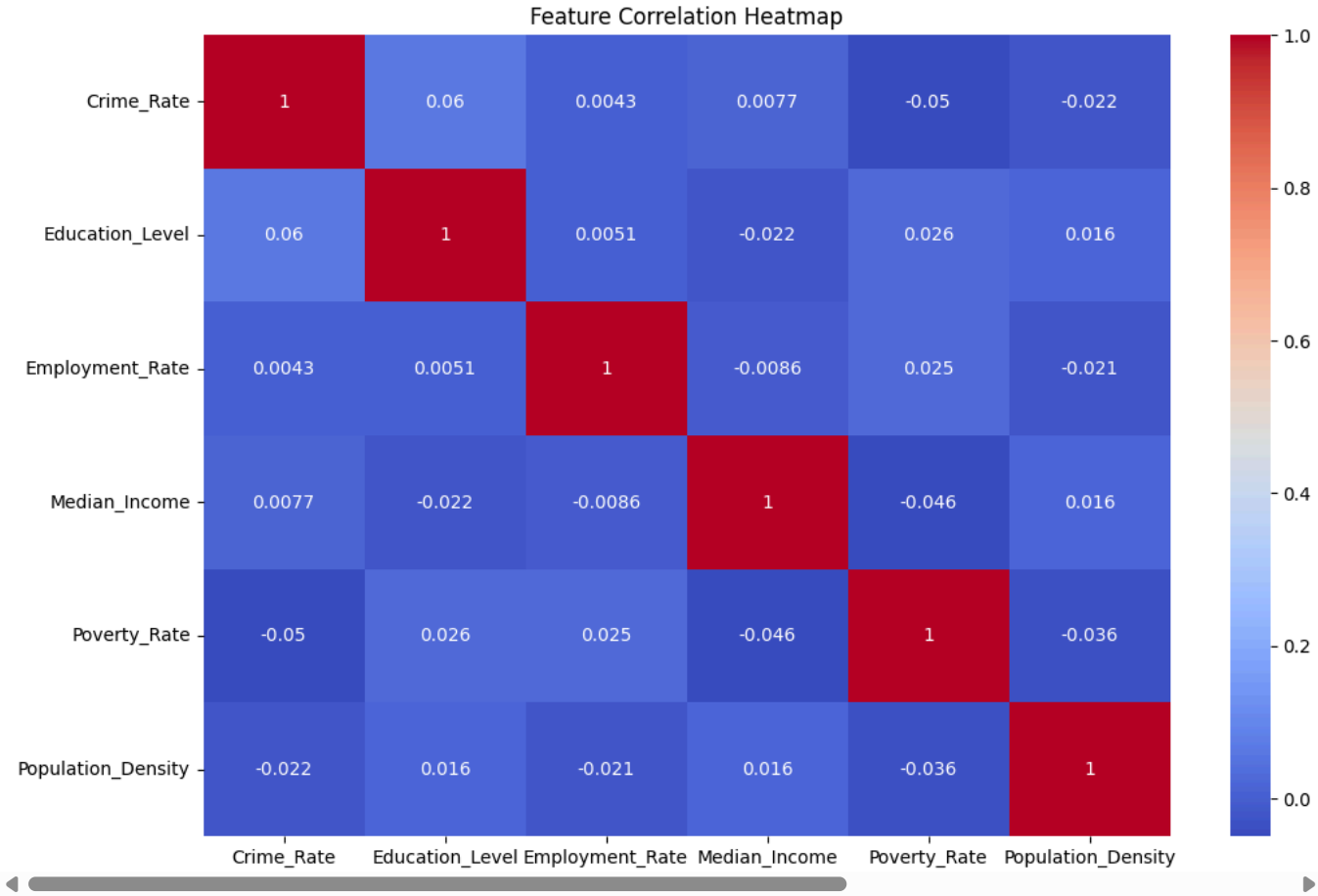
Visualize a Few Features

```
import seaborn as sns
import matplotlib.pyplot as plt

# Example: visualizing 'Crime_Rate' distribution
sns.histplot(df['Crime_Rate'], kde=True)
plt.title('Distribution of Crime Rate')
plt.xlabel('Crime Rate')
plt.show()
```




```
# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```



Identify Target and Features

```
target = 'Crime_Rate'  # Make sure this is the correct column name
features = df.columns.drop(target)
print("Features:", features.tolist())
```

```
Features: ['Region', 'Education_Level', 'Employment_Rate', 'Median_Income', 'Poverty_Rate', 'Population_Density']
```

Convert Categorical Columns to Numerical

```
categorical_cols = df.select_dtypes(include=['object']).columns
print("Categorical Columns:", categorical_cols.tolist())
```

```
Categorical Columns: ['Region']
```

One-Hot Encoding

```
df_encoded = pd.get_dummies(df, drop_first=True)
X = df_encoded.drop(columns=[target])
y = df_encoded[target]
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded.drop(target, axis=1))
y = df_encoded[target]
```

Train-Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Model Building

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

Evaluation

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)
print("MSE:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
MSE: 199267.91585000002
R² Score: -0.11007296536141076
```

Make Predictions from New Input

```
# Replace with actual values from your dataset's features
new_input = {
    'feature1': 0.25,
    'feature2': 1,
    'feature3': 3.5,
    # Add all features here based on df.columns.drop('Crime_Rate')
}
```

Convert to DataFrame and Encode

```
new_df = pd.DataFrame([new_input])
df_temp = pd.concat([df.drop(target, axis=1), new_df], ignore_index=True)
df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)
df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop(target, axis=1).columns, fill_value=0)

new_input_scaled = scaler.transform(df_temp_encoded.tail(1))
```

Predict the Final Grade (in this case: Crime Rate)

```
predicted_crime_rate = model.predict(new_input_scaled)
print("🚨 Predicted Crime Rate:", round(predicted_crime_rate[0], 2))
```

```
🚨 Predicted Crime Rate: 813.66
```

Deployment using Gradio

```
!pip install gradio
```

```
Requirement already satisfied: gradio in /usr/local/lib/python3.11/dist-packages (5.29.1)
Requirement already satisfied: aiofiles<25.0,>=22.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (24.1.0)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.115.12)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.11/dist-packages (from gradio) (0.5.0)
Requirement already satisfied: gradio-client==1.10.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (1.10.1)
Requirement already satisfied: groovy~=0.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.1.2)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.31.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.18)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.4)
Requirement already satisfied: pydub in /usr/local/lib/python3.11/dist-packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Requirement already satisfied: ruff>=0.9.3 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.11.10)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.1.6)
Requirement already satisfied: semantic-version~=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.46.2)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.13.2)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Requirement already satisfied: uvicorn>=0.14.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.34.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.1->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.1->gradic
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.1
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradic
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.2.0)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->g
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0.1
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28.1
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typ
```

```
print(df['Crime_Rate'].unique())
```

```
[1176  910 1344 1180 1145 1094  171  516 1288  380  137 1446 1173  921
 180 1382  819  393 1487  855  435 1265 1005  326 1234  509 1387   71
 302  797  906  524 1132  560  749 1025  239 1007  736  612 1317  881
1204  696   70  890  216 1347  437  650  365   63  291  826 1419  614
 947 1413  141 1440  558  825   84  255 1154 1461 1075 1071 1463  615
1179  752  451  779  211  251 1045  319  865  505 1325 1066  345  769
 387  928 1126  841  266  813  237  429  542 1114 1230   64  114  570
1417 1202  697 1136 1212  642  441  468  338  428  280 1067   90 1101
 184  250  829  979 1106 1121  552  456  854  148  733  775 1010  662
 692 1078 1291  816  447  920  844  442  256 1088  941  913  792 1314
 613 1070  145  534 1480  798 1244  590 1109  574 1233 1316 1159  627
 731  606 1062  695  845 1404 1421 1304 1298 1458 1450 1356  682  677
1022 1332 1432  759 1484  698  367 1307  713 1447 1306 1186  491 1341
1459 1460 1001  748  162   51  691  269  904  434 1476 1203 1489  296
 885  252 1257  172  450  343  329  933  247 1031  801  193  658 1274
1197  236 1399  513 1422  709 1004  452  196 1221  538  978  600  690
 522  200 1488 1371  312  395  673  930  946 1377  303  502 1110 1064
  58 1172  257  453  201  103 1193  636 1493 1495  153 1300 1185  559
 202  963  927  871 1236 1006  210  850 1350 1465 1208 1268  689  737
 519 1448 1095  799   87  279 1438  487   76  275  847  333 1009 1069
 708  596 1122   66 1245 1293  207  526 1119  567  546  351  230  656
1042 1189  240 1030  977 1329 1396  177   67  619  735  675 1361  903
1011  204  539  834 1177 1466  295 1249 1112  808  587  970  457  877
 952   85  734   69  370  561  449  703 1021  520 1216 1165 1427  883
 776  100  624 1263 1198  199  363 1415  354  887 1127 1052  493  313
1053   54 1269  823  926 1041 1423   96  916  872  985  705  318  419
 685  706  169  880 1131  790  166  840 1200  683  562  851 1169 1191
 609  537 1310 1345 1262  753  630  839 1149  203 1046  746 1358 1142
1120 1058  143 1061 1277  961  267  523  481  390 1362 1327  406   72
```

```
 571 1253 1296  811  818 1065 1398  565 1089  385  307  945  209  901
 739  740 1490 1241 1116  334  446   61 1375 1326  548  803  898  139
 654 1019  467  164  666  875 1148  462 1449  778  810  725 1467  506
 969  241  788 1027  712   86 1442  328  922 1346  111  645 1092  226
 445  494  125  314  504  767  784 1457  900  934 1098  997  940 1224
1217  618  158 1115 1259  984 1206  782  284  707  800  637 1082 1147
1003  541  962 1303 1247  219  995 1267   78 1238  388  549 1400 1374
1205 1143  925 1255  140  763   88 1199  222  702  525  878  944 1434
1074  436  397  554 1264 1482  873  983 1190  183  107  721  864 1222
 129  935 1286 1302 1475 1378 1020 1085 1403  785  833 1017  407 1481
 717  422 1081 1195  747 1389 1251  589  781  918  789  551 1008 1218
 510  974  958  439  932  305  758  499   59 1315 1324 1192  423 1138
 195  273 1312  902 1355  112 1290  876  738  154  454  820 1096  726
 132 1158 1151   50  766  498   83  144  299  723  167 1379  317  285
 438 1039  812 1376  968 1280  124  484  152  787  147 1408  908  496
1381  298  215 1033  591  130  182  846  949  187  233 1474  118  595
 102  254  175   92  160 1256  897  409  330  716  142  110 1124  653
 583 1478  694  282  573 1146  843 1194  263  288 1129  399  156   97
 386 1099  853 1330   57  674  768  352  688  105  575   77  127 1289
 414  715 1385 1080 1076  412  135  861  586  194 1285   74  116 1219
 277  672  644  474  741 1393  364  711  892 1210  948  292  592 1113
 278  597   55 1395  517   53  553  996  623  243  896  764 1373  375
1483  773  274  581]
```

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(random_state=42)  # this RE-DEFINES the trained model
```

```
import gradio as gr
import numpy as np

# Get feature names from X used earlier
feature_names = X.columns.tolist()

def predict_crime_rate(*inputs):
    input_array = np.array(inputs).reshape(1, -1)
    scaled_input = scaler.transform(input_array)
    prediction = model.predict(scaled_input)[0]
    return round(prediction, 2)


input_components = [gr.Number(label=feature) for feature in feature_names]


# Launch the app
gr.Interface(
    fn=predict_crime_rate,
    inputs=input_components,
    outputs=gr.Number(label="Predicted Crime Rate"),
    title="🚨 Crime Rate Prediction App",
    description="Enter socioeconomic factors to predict the crime rate using a trained machine learning model."
).launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatica

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://ebb27c04bd336c4ae0.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the workin
```