```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df=pd.read_csv("/content/drive/MyDrive/Fraud.csv")
```

DATA PREPROCESSING

```
df1=df
df.head(10)
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDes
0	1	PAYMENT	9839.64	C1231006815	170136.00	160296.36	M197978715
1	1	PAYMENT	1864.28	C1666544295	21249.00	19384.72	M204428222
2	1	TRANSFER	181.00	C1305486145	181.00	0.00	C55326406
3	1	CASH_OUT	181.00	C840083671	181.00	0.00	C3899701
4	1	PAYMENT	11668.14	C2048537720	41554.00	29885.86	M123070170
5	1	PAYMENT	7817.71	C90045638	53860.00	46042.29	M57348727
6	1	PAYMENT	7107.77	C154988899	183195.00	176087.23	M40806911
7	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M63332633
8	1	PAYMENT	4024.36	C1265012928	2671.00	0.00	M11769321C
9	1	DEBIT	5337.77	C712410124	41720.00	36382.23	C19560086

```
df.shape
```

```
(6362620, 11)
```

```
df.dtypes
```

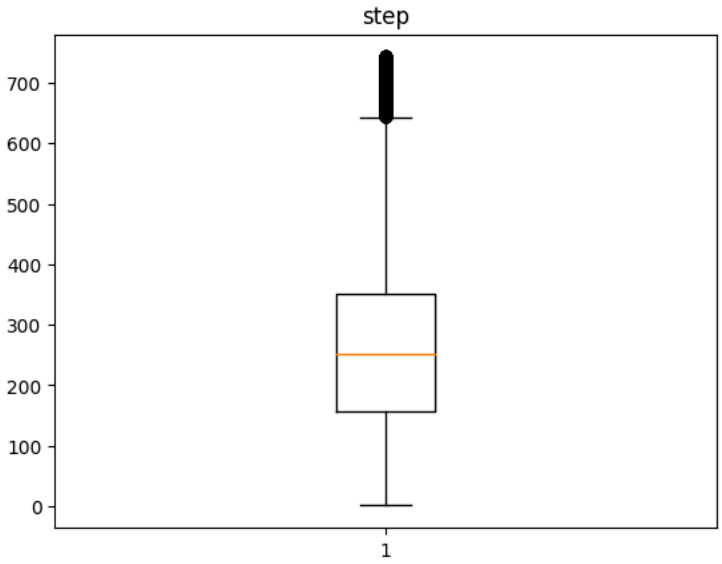
```
step              int64
type             object
amount          float64
nameOrig         object
oldbalanceOrg   float64
newbalanceOrig  float64
nameDest         object
oldbalanceDest  float64
newbalanceDest  float64
isFraud           int64
isFlaggedFraud    int64
dtype: object
```

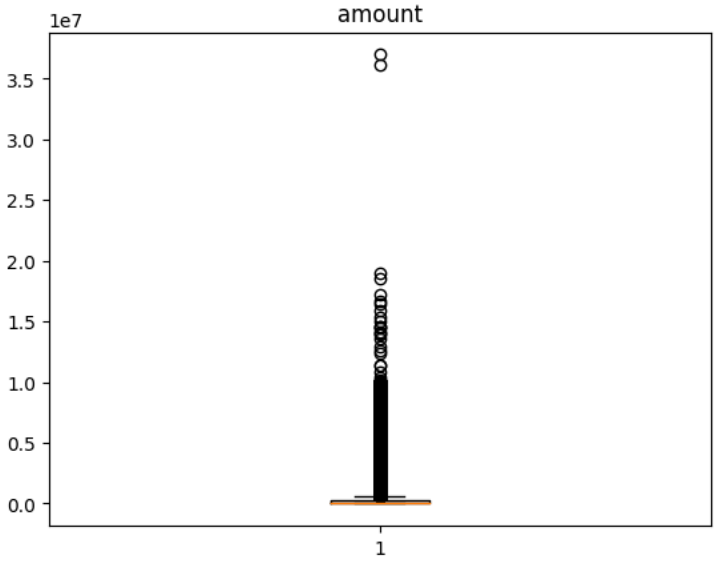
```
df.isnull().sum()
```

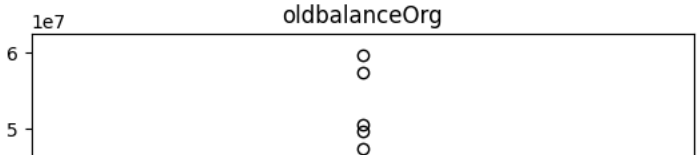
```
step            0
type            0
amount          0
nameOrig        0
oldbalanceOrg   0
newbalanceOrig  0
nameDest        0
oldbalanceDest  0
newbalanceDest  0
isFraud         0
isFlaggedFraud  0
dtype: int64
```

Checking for outliers using box-plot

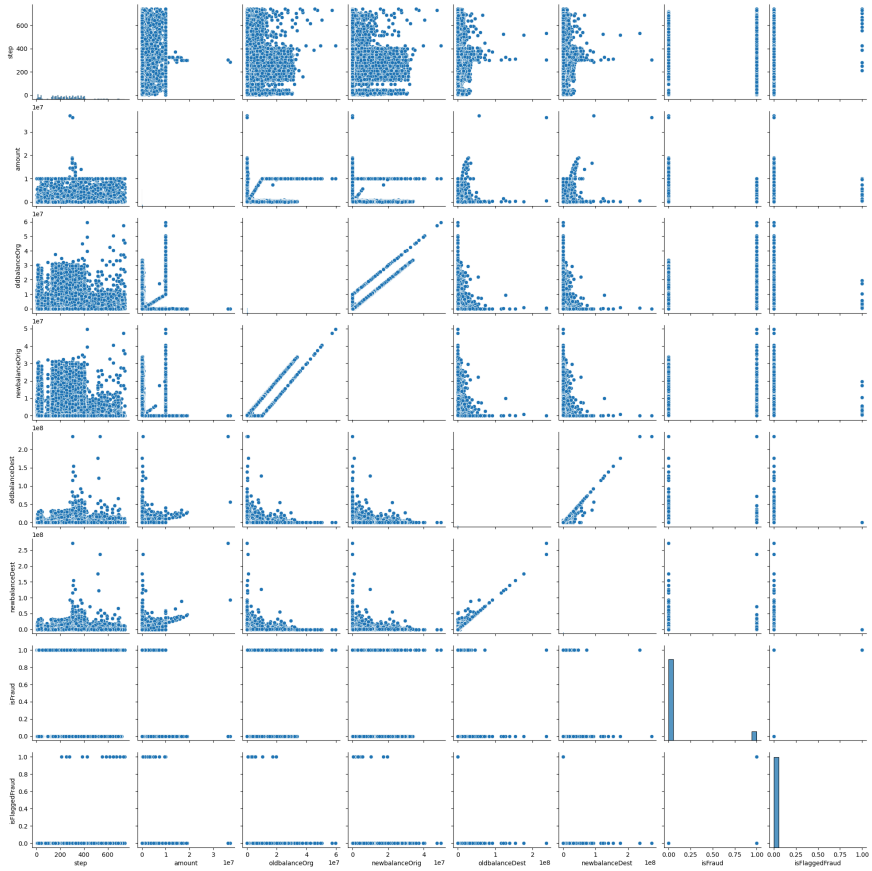
```
l=df.columns
for x in l:
    if df[x].dtype=="int64" or df[x].dtype=="float64":
        plt.boxplot(df[x])
        plt.title(x)
        plt.show()
```








```
import seaborn as sns
sns.pairplot(df)
plt.show()
```




```
def detect_outliers(series):
    q1=series.quantile(0.25)
    q3=series.quantile(0.75)
    iqr=q3-q1
    lower_bound=q1-1.5*iqr
    upper_bound=q3+1.5*iqr
    outliers=(series<lower_bound)|(series>upper_bound)
    return outliers.sum()
for x in df.columns:
    if df[x].dtype=="int64" or df[x].dtype=="float64":
        num_outliers = detect_outliers(df[x])
        print(f"Number of outliers in column '{x}': {num_outliers}")
```

```
Number of outliers in column 'step': 102688
Number of outliers in column 'amount': 338078
Number of outliers in column 'oldbalanceOrg': 1112507
Number of outliers in column 'newbalanceOrig': 1053391
Number of outliers in column 'oldbalanceDest': 786135
Number of outliers in column 'newbalanceDest': 738527
Number of outliers in column 'isFraud': 8213
Number of outliers in column 'isFlaggedFraud': 16
```

```
df.type.unique()
```

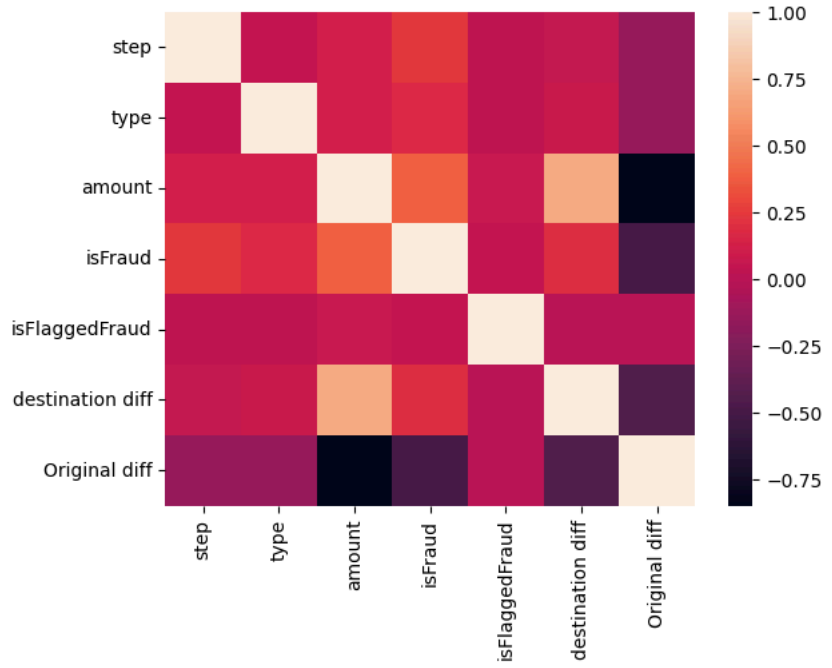
```
array(['PAYMENT', 'TRANSFER', 'CASH_OUT', 'DEBIT', 'CASH_IN'],
      dtype=object)
```

```
from sklearn import preprocessing
le=preprocessing.LabelEncoder()
df1['type']=le.fit_transform(df1['type'])


df1['destination diff']=df1.newbalanceDest-df1.oldbalanceDest
df1['Original diff']=df1.newbalanceOrig-df1.oldbalanceOrg


del df1["newbalanceDest"]
del df1["oldbalanceDest"]
del df1["newbalanceOrig"]
del df1["oldbalanceOrg"]



del df1['nameDest']
del df1['nameOrig']


df.corr()
```

	step	type	amount	isFraud	isFlaggedFraud	destination diff
step	1.000000	0.006635	0.022373	0.031578	0.003277	0.001325
type	0.006635	1.000000	0.088419	0.020833	0.002685	0.169399
amount	0.022373	0.088419	1.000000	0.076688	0.012295	0.845964
isFraud	0.031578	0.020833	0.076688	1.000000	0.044109	0.027028
isFlaggedFraud	0.003277	0.002685	0.012295	0.044109	1.000000	-0.000242
destination diff	0.001325	0.169399	0.845964	0.027028	-0.000242	1.000000

```
import seaborn as sns
sns.heatmap(df.corr())
```

```
<Axes: >
```




```
df.head()
```

	step	type	amount	isFraud	isFlaggedFraud	destination diff	Original diff
0	1	3	9839.64	0	0	0.0	-9839.64
1	1	3	1864.28	0	0	0.0	-1864.28
2	1	4	181.00	1	0	0.0	-181.00
3	1	1	181.00	1	0	-21182.0	-181.00
4	1	3	11668.14	0	0	0.0	-11668.14

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report


# since there are many outliers in the important features, I have tried using three ML models which are not as sensitive
# compared to other models. Removing outliers in cases where there are binary classes will affect the entire prediction
# and they cannot be exchanged with the mean,median or mode.


independent=["type","amount","destination diff","Original diff"]

# I have selected the above features for prediction because name of the source/destination does not affect the decision
# of a transaction being fraud or not. Features like total amount transacted, and the resultant balance in original account
# will allow us to understand if its a fraud or not. Even the difference in the balance of the destination account. And many
# times there are cases where the transaction type and how long it took leads to understanding if its a fraud or not. Eg:
# if a transaction is taking long then the destination must be well hidden.

df_subset = df1.sample(frac=0.1, random_state=42)

# due to large data size, I have tried random sampling to see how it works. Due to need of more storage I shifted to kaggle
# to use its storage and GPU power. The accuracy remains the same even when applying it to the entire dataset.

X_subset = df_subset[independent]
y_subset = df_subset["isFraud"]
X_train_subset, X_test_subset, y_train_subset, y_test_subset = train_test_split(
    X_subset, y_subset, test_size=0.2, random_state=42)


rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_subset, y_train_subset)
rf_predictions = rf_model.predict(X_test_subset)


print("Random Forest:")
print("Accuracy:", accuracy_score(y_test_subset, rf_predictions))
print("Classification Report:\n", classification_report(y_test_subset, rf_predictions))
```

```
Random Forest:
Accuracy: 0.9993634727668503
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    127079
           1       0.80      0.71      0.75       174

    accuracy                           1.00    127253
   macro avg       0.90      0.86      0.88    127253
weighted avg       1.00      1.00      1.00    127253
```

```
import xgboost as xgb

xgb_model = xgb.XGBClassifier(n_estimators=100, random_state=42)
xgb_model.fit(X_train_subset, y_train_subset)
xgb_predictions = xgb_model.predict(X_test_subset)

print("\nXGBoost:")
print("Accuracy:", accuracy_score(y_test_subset, xgb_predictions))
print("Classification Report:\n", classification_report(y_test_subset, xgb_predictions))

#there are 100 decision trees, based on n_estimators value
for x in range(10):
  plt.figure(figsize=(20, 10))
  xgb.plot_tree(xgb_model, num_trees=x)
  plt.show()
```

<Figure size 2000x1000 with 0 Axes>

<Figure size 2000x1000 with 0 Axes>

<Figure size 2000x1000 with 0 Axes>

<Figure size 2000x1000 with 0 Axes>

<Figure size 2000x1000 with 0 Axes>

<Figure size 2000x1000 with 0 Axes>

<Figure size 2000x1000 with 0 Axes>

```
import json
import numpy as np

l=[4, 10101, 0, -10101]
a=np.array(l)
output=xgb_model.predict(a.reshape(1,-1))
my_number = np.int64(output[0])
json_data = json.dumps(int(my_number))  # Convert to int before JSON serialization
print(json_data)
```

```
1
```

```
svm_model = SVC(random_state=42)
svm_model.fit(X_train_subset, y_train_subset)
svm_predictions = svm_model.predict(X_test_subset)

print("\nSVM:")
print("Accuracy:", accuracy_score(y_test_subset, svm_predictions))
print("Classification Report:\n", classification_report(y_test_subset, svm_predictions))
```

```
SVM:
Accuracy: 0.9651651651651652
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1512
           1       0.93      0.67      0.78       153

    accuracy                           0.97      1665
   macro avg       0.95      0.83      0.88      1665
weighted avg       0.96      0.97      0.96      1665
```

```
!pip install gradio
```

```
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.3)
Requirement already satisfied: markupsafe~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.1.5)
Requirement already satisfied: matplotlib~=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Requirement already satisfied: numpy~=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.25.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.10.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (24.0)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.0.3)
Requirement already satisfied: pillow<11.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (9.4.0)
Requirement already satisfied: pydantic>=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.7.0)
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.9 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.0.9)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.1)
Requirement already satisfied: ruff>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.4.2)
Requirement already satisfied: semantic-version~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.10.0)
Requirement already satisfied: tomlkit==0.12.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.12.0)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.12.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.11.0)
Requirement already satisfied: urllib3~=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.0.7)
Requirement already satisfied: uvicorn>=0.14.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.29.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.16.0->gradio) (2023.6.0)
Requirement already satisfied: websockets<12.0,>=10.0 in /usr/local/lib/python3.10/dist-packages (from gradio-client==0.16.0->gra
```

```
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.3->gradio) (3.13.4
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.3->gradio) (2.31.0
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.19.3->gradio) (4.
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (4.51.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (1.4.5
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib~=3.0->gradio) (2.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2024.1)
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (0.
Requirement already satisfied: pydantic-core==2.18.1 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (2.1
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.7)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (1.5
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (13.7.1)
Requirement already satisfied: starlette<0.38.0,>=0.37.2 in /usr/local/lib/python3.10/dist-packages (from fastapi->gradio) (0.37.
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.0
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair<6.0,>=4.2.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib~=3.0->g
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<1.0,>=
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich>=10.11.0->typer<1.0,
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio->httpx>=0.24.1->gradio) (1.2
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hu
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->
```

```
import gradio as gr
import json
import numpy as np

def greet(Transaction_type, amount, destinationdiff, Originaldiff):
  d={"TRANSFER": 4, "CASH_IN": 0, "CASH_OUT": 1, "PAYMENT": 3, "DEBIT": 2}
  l=[d[Transaction_type], amount, destinationdiff, Originaldiff]
  a=np.array(l)
  output=xgb_model.predict(a.reshape(1,-1))
  my_number = np.int64(output[0])
  json_data = json.dumps(int(my_number))
  return json_data

demo = gr.Interface(
    greet,
    [
         gr.Radio(["TRANSFER", "CASH_IN", "CASH_OUT", "PAYMENT", "DEBIT"]),
         "number",
         "number",
         "number"
    ],
    "number",
    title="Transaction Details"
)

demo.launch()
```

```
/usr/local/lib/python3.10/dist-packages/gradio/analytics.py:99: UserWarning: unable to parse version details from package URL.
  warnings.warn("unable to parse version details from package URL.")
Setting queue=True in a Colab notebook requires sharing enabled. Setting `share=True` (you can turn this off by setting `share=False

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()

Could not create share link. Missing file: /usr/local/lib/python3.10/dist-packages/gradio/frpc_linux_amd64_v0.2.

Please check your internet connection. This can happen if your antivirus software blocks the download of this file. You can install

1. Download this file: https://cdn-media.huggingface.co/frpc-gradio-0.2/frpc_linux_amd64
2. Rename the downloaded file to: frpc_linux_amd64_v0.2
3. Move the file to this location: /usr/local/lib/python3.10/dist-packages/gradio
Running on https://localhost:7863/
```