Fast Tag Fraud Detection

Author: Irfan Ullah Khan

GITHUB PROFILE | KAGGLE PROFILE | LINKEDIN PROFILE

YOUTUBE PROFILE | EMAIL CONTACT ME | WEBSITE CONTACT ME

**Load Libraries **

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report
import tensorflow as tf
```

Load Dataset

```
df = pd.read_csv('/content/FastagFraudDetection.csv')
```

```
df.head()
```

	Transaction_ID	Timestamp	Vehicle_Type	FastagID	TollBoothID	Lane_Type	Vehi
0	1	01-06-23 11:20	Bus	FTG-001-ABC-121	A-101	Express	
1	2	01-07-23 14:55	Car	FTG-002-XYZ-451	B-102	Regular	
2	3	01-08-23 18:25	Motorcycle	NaN	D-104	Regular	
3	4	01-09-23 2:05	Truck	FTG-044-LMN-322	C-103	Regular	
4	5	01-10-23 6:35	Van	FTG-505-DEF-652	B-102	Express	

Next steps: Generate code with df | View recommended plots

```
df.head(20)
```

	Transaction_ID	Timestamp	Vehicle_Type	FastagID	TollBoothID	Lane_Type	Veh
0	1	01-06-23 11:20	Bus	FTG-001-ABC-121	A-101	Express	
1	2	01-07-23 14:55	Car	FTG-002-XYZ-451	B-102	Regular	
2	3	01-08-23 18:25	Motorcycle	NaN	D-104	Regular	
3	4	01-09-23 2:05	Truck	FTG-044-LMN-322	C-103	Regular	
4	5	01-10-23 6:35	Van	FTG-505-DEF-652	B-102	Express	
5	6	01-11-23 10:00	Sedan	FTG-066-GHI-987	A-101	Regular	
6	7	01-12-23 15:40	SUV	FTG-707-JKL-210	B-102	Express	
7	8	1/13/2023 20:15	Bus	FTG-088-UVW-543	C-103	Regular	
8	9	1/14/2023 1:55	Car	FTG-909-RST-876	A-101	Express	
		1/15/2023					

9	10	1/15/2023 7:30	Motorcycle	NaN	D-104	Regular
10	11	1/16/2023	Truck	FTG-021-	C-103	Express

Next steps: Generate code with df View recommended plots

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Transaction_ID         5000 non-null   int64
 1   Timestamp              5000 non-null   object
 2   Vehicle_Type           5000 non-null   object
 3   FastagID               4451 non-null   object
 4   TollBoothID            5000 non-null   object
 5   Lane_Type              5000 non-null   object
 6   Vehicle_Dimensions     5000 non-null   object
 7   Transaction_Amount     5000 non-null   int64
 8   Amount_paid            5000 non-null   int64
 9   Geographical_Location  5000 non-null   object
 10  Vehicle_Speed          5000 non-null   int64
 11  Vehicle_Plate_Number   5000 non-null   object
 12  Fraud_indicator        5000 non-null   object
dtypes: int64(4), object(9)
memory usage: 507.9+ KB
```

```
df.describe()
```

	Transaction_ID	Transaction_Amount	Amount_paid	Vehicle_Speed
count	5000.000000	5000.00000	5000.000000	5000.000000
mean	2500.500000	161.06200	141.261000	67.851200
std	1443.520003	112.44995	106.480996	16.597547
min	1.000000	0.00000	0.000000	10.000000
25%	1250.750000	100.00000	90.000000	54.000000
50%	2500.500000	130.00000	120.000000	67.000000
75%	3750.250000	290.00000	160.000000	82.000000
max	5000.000000	350.00000	350.000000	118.000000

```
df.shape
```

```
(5000, 13)
```

Convert 'Timestamp' column to datetime

```
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
```

```
df.isnull()
```

	Transaction_ID	Timestamp	Vehicle_Type	FastagID	TollBoothID	Lane_Type	V
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	True	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
4995	False	False	False	False	False	False	
4996	False	False	False	False	False	False	
4997	False	False	False	False	False	False	
4998	False	False	False	False	False	False	
4999	False	False	False	False	False	False	

5000 rows × 13 columns

```
print("Missing values in 'FastagID':", df['FastagID'].isnull().sum())
```

```
Missing values in 'FastagID': 549
```

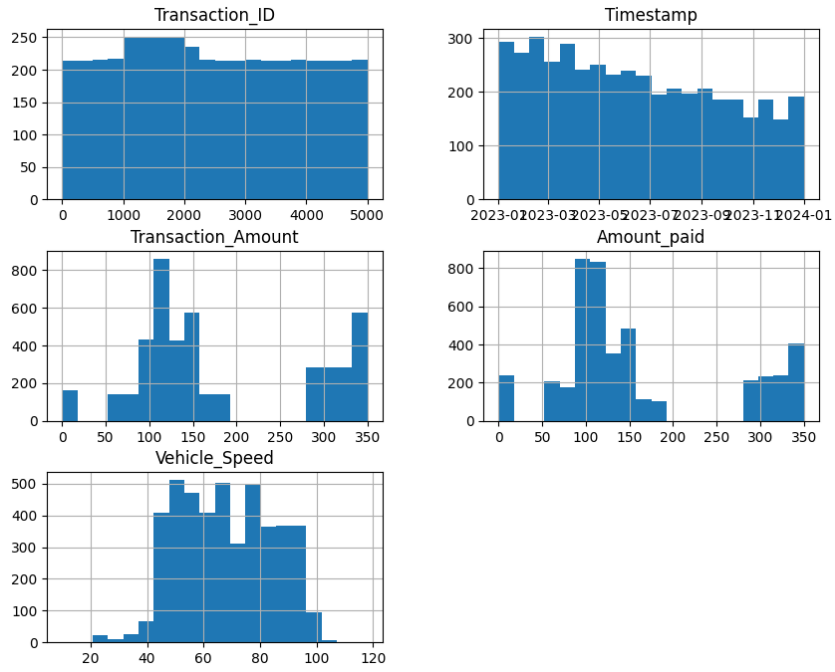
```
df = df.dropna(subset=['FastagID'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4451 entries, 0 to 4999
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Transaction_ID         4451 non-null   int64
 1   Timestamp              4451 non-null   datetime64[ns]
 2   Vehicle_Type           4451 non-null   object
 3   FastagID               4451 non-null   object
 4   TollBoothID            4451 non-null   object
 5   Lane_Type              4451 non-null   object
 6   Vehicle_Dimensions     4451 non-null   object
 7   Transaction_Amount     4451 non-null   int64
 8   Amount_paid            4451 non-null   int64
 9   Geographical_Location  4451 non-null   object
 10  Vehicle_Speed          4451 non-null   int64
 11  Vehicle_Plate_Number   4451 non-null   object
 12  Fraud_indicator        4451 non-null   object
dtypes: datetime64[ns](1), int64(4), object(8)
memory usage: 486.8+ KB
```

Histogram

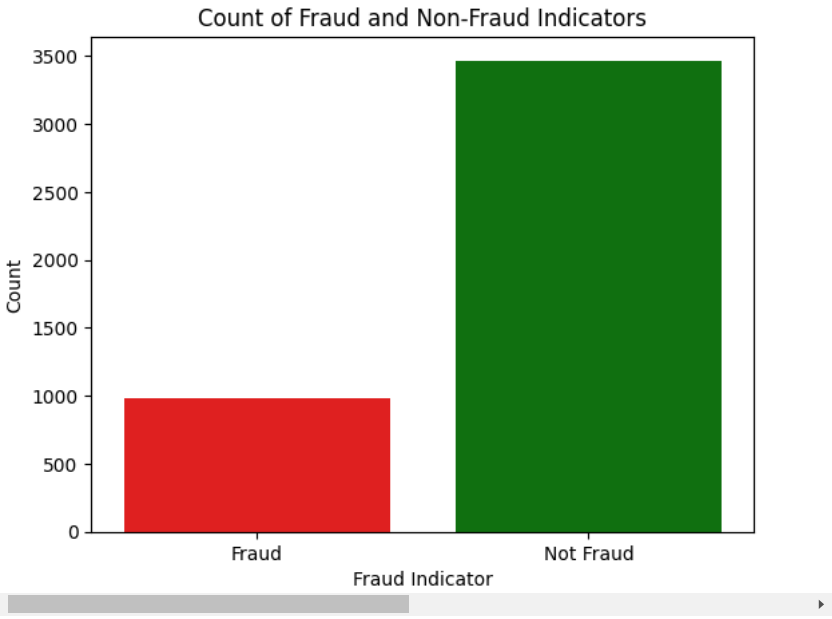
```
df.hist(figsize=(10, 8), bins=20)
plt.show()
```



Count of Fraud and Non_Fraud Indicators bold text

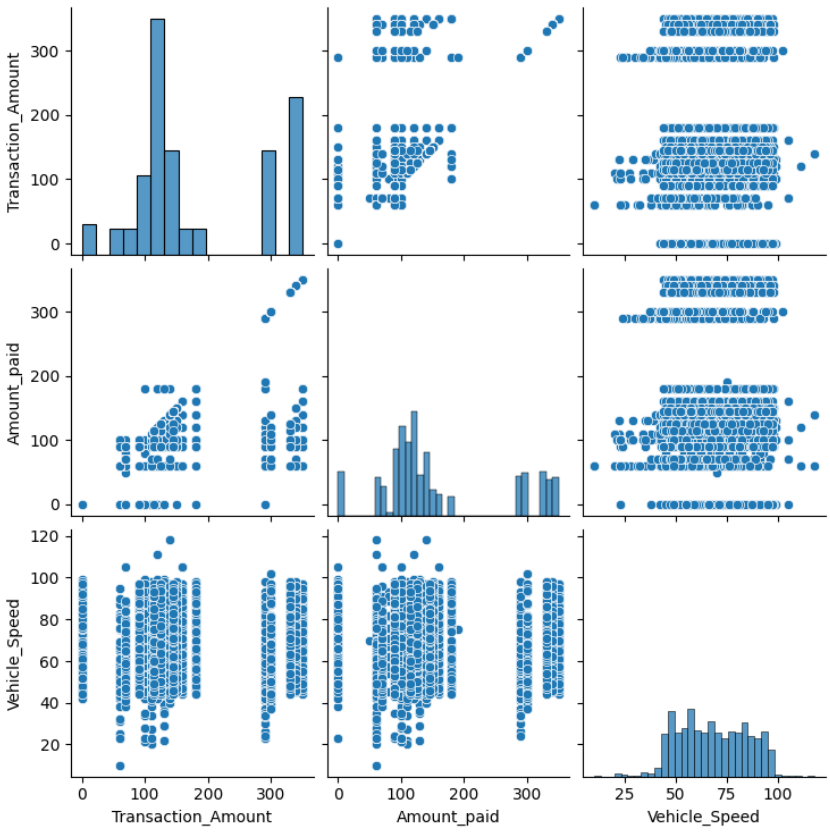
```
sns.countplot(x='Fraud_indicator', data=df, palette=['red', 'green'])
plt.xlabel('Fraud Indicator')
plt.ylabel('Count')
plt.title('Count of Fraud and Non-Fraud Indicators')
plt.show()
```

```
<ipython-input-14-a0b8bf54ff9b>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.

  sns.countplot(x='Fraud_indicator', data=df, palette=['red', 'green'])
```



Pairwise scatter plots for numerical variables

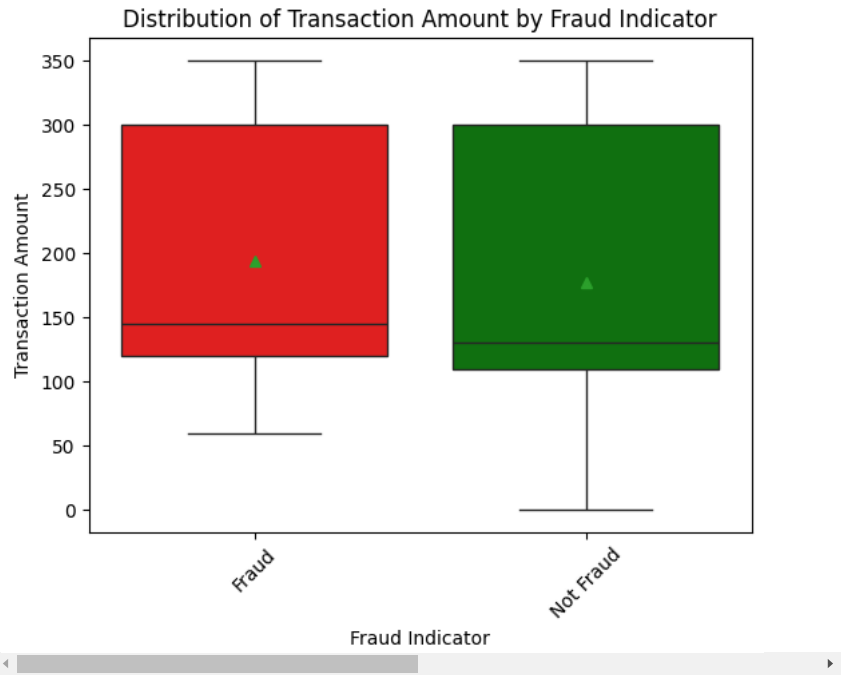
```
sns.pairplot(df, vars=['Transaction_Amount', 'Amount_paid', 'Vehicle_Speed'])
plt.show()
```



Box plot for 'Transaction_Amount' and 'Amount_paid'

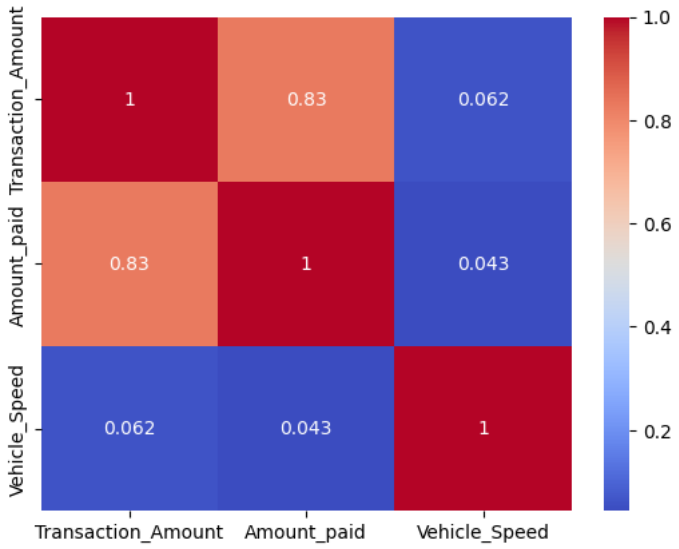
```
sns.boxplot(
    x = "Fraud_indicator",
    y = "Transaction_Amount",
    showmeans=True,
    data=df,
    palette=["red", "green"]
)

plt.xlabel("Fraud Indicator")
plt.ylabel("Transaction Amount")
plt.title("Distribution of Transaction Amount by Fraud Indicator")
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-16-b6b2f11221a9>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.

  sns.boxplot(
```



Correlation matrix and heatmap for numerical variables

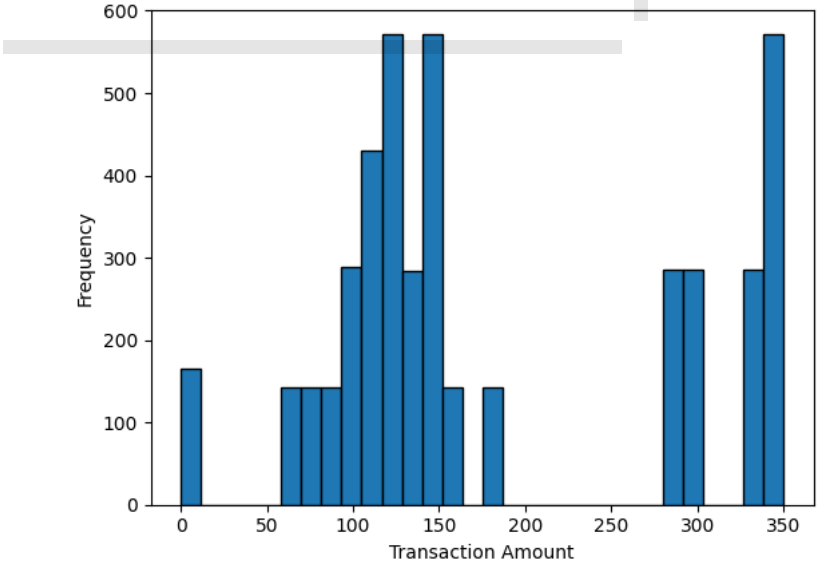
```
correlation_matrix = df[['Transaction_Amount', 'Amount_paid', 'Vehicle_Speed']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```



Histogram of 'Transaction_Amount'

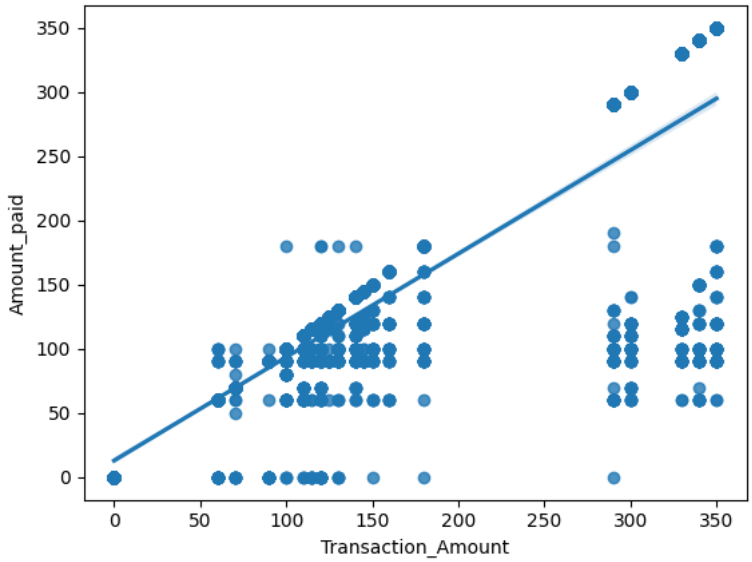
```
plt.hist(df['Transaction_Amount'], bins=30, edgecolor='black'
plt.xlabel('Transaction Amount')
```

```
plt.ylabel('Frequency')
```

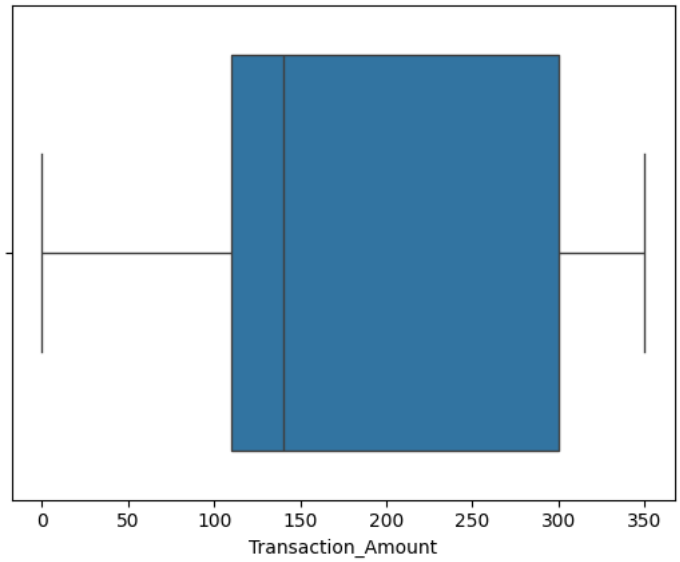


Scatter plot with regression line between 'Transaction_Amount' and 'Amount_paid'

```
sns.regplot(x='Transaction_Amount', y='Amount_paid', data=df)
plt.show()
```




```
sns.boxplot(x='Transaction_Amount', data=df)
plt.show()
```



Select features Transaction_Amount, Amount_paid

```
selected_features = ['Transaction_Amount', 'Amount_paid']
X = df[selected_features]
y = df['Fraud_indicator']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Scaling and encoding output

```
from sklearn.preprocessing import LabelEncoder
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

Neural network model

```
from tensorflow.keras import models, layers

model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(X_train_scaled.shape[1],)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
model.summary()
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                96

 dense_1 (Dense)             (None, 16)                528

 dense_2 (Dense)             (None, 1)                 17

=================================================================
Total params: 641 (2.50 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.fit(X_train_scaled, y_train_encoded, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
89/89 [==============================] - 1s 5ms/step - loss: 0.4898 - accuracy: 0.8213 - val_loss: 0.4044 - val_accuracy: 0.8371
Epoch 2/10
89/89 [==============================] - 0s 3ms/step - loss: 0.3257 - accuracy: 0.8627 - val_loss: 0.2700 - val_accuracy: 0.8919
Epoch 3/10
89/89 [==============================] - 0s 3ms/step - loss: 0.2149 - accuracy: 0.9171 - val_loss: 0.1833 - val_accuracy: 0.9312
Epoch 4/10
```

```
89/89 [==============================] - 0s 3ms/step - loss: 0.1523 - accuracy: 0.9459 - val_loss: 0.1428 - val_accuracy: 0.9607
Epoch 5/10
89/89 [==============================] - 0s 3ms/step - loss: 0.1185 - accuracy: 0.9565 - val_loss: 0.1182 - val_accuracy: 0.9803
Epoch 6/10
89/89 [==============================] - 0s 3ms/step - loss: 0.0981 - accuracy: 0.9775 - val_loss: 0.1039 - val_accuracy: 0.9817
Epoch 7/10
89/89 [==============================] - 0s 3ms/step - loss: 0.0855 - accuracy: 0.9814 - val_loss: 0.0959 - val_accuracy: 0.9817
Epoch 8/10
89/89 [==============================] - 0s 3ms/step - loss: 0.0762 - accuracy: 0.9828 - val_loss: 0.0901 - val_accuracy: 0.9817
Epoch 9/10
89/89 [==============================] - 0s 3ms/step - loss: 0.0692 - accuracy: 0.9838 - val_loss: 0.0849 - val_accuracy: 0.9817
Epoch 10/10
89/89 [==============================] - 0s 3ms/step - loss: 0.0645 - accuracy: 0.9838 - val_loss: 0.0815 - val_accuracy: 0.9817
<keras.src.callbacks.History at 0x7fb7681cd630>
```

Print accuracy metrics

```
y_pred_prob = model.predict(X_test_scaled)
# Convert probabilities to binary predictions
y_pred = np.round(y_pred_prob)
# Print accuracy metrics
accuracy = accuracy_score(y_test_encoded, y_pred)
precision = precision_score(y_test_encoded, y_pred)
recall = recall_score(y_test_encoded, y_pred)
f1 = f1_score(y_test_encoded, y_pred)
# Print accuracy metrics
print("Accuracy: {:.2f}%".format(accuracy * 100))
print("Precision: {:.2f}%".format(precision * 100))
print("Recall: {:.2f}%".format(recall * 100))
print("F1 Score: {:.2f}%".format(f1 * 100))
```

```
28/28 [==============================] - 0s 1ms/step
Accuracy: 98.43%
Precision: 98.04%
Recall: 100.00%
F1 Score: 99.01%
```

Accuracy metrics graph

```
import matplotlib.pyplot as plt


metrics = ['Accuracy', 'Precision', 'Recall', 'F1 Score']
values = [99.66 ,99.57, 100.00 , 99.79]

plt.bar(metrics, values, color=['blue', 'green', 'orange', 'red'])
plt.ylabel('Score')
plt.title('Model Metrics')
plt.ylim(0, 1)
plt.show()
```