Out[

FAKE BILLS PREDICTION

In [41]:

```
import numpy as np import pandas as pd import seaborn as sns
import matplotlib.pyplot as plt from sklearn.model_selection import
train_test_split from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score from sklearn.preprocessing
import StandardScaler
from sklearn.metrics import accuracy_score,classification_report,confusion_mat
```

In [13]:

```
df = pd.read_excel("fake_bills_converted.xlsx")
```

In [14]:

```
df
```

Out[14]:

```
                         is_genuine diagonal height_left height_right margin_low margin_up le

        0        1    171.81     104.86     104.95       4.52        2.89 11
11      1        1    171.46     103.36     103.66       3.77        2.99
11      2        1    172.69     104.48     103.50       4.40        2.94
        3        1    171.36     103.91     103.94       3.62        3.01
                                                                          11
        4        1    171.73     104.28     103.46       4.04        3.48  11
                ...      ...        ...        ...        ...         ...
     1495        0    171.75     104.38     104.17       4.42        3.09  11
     1496        0    172.19     104.63     104.44       5.27        3.37  11
     1497        0    171.80     104.01     104.12       5.51        3.36  11
     1498        0    172.06     104.28     104.06       5.17        3.46  11
     1499        0    171.47     104.15     103.82       4.63        3.37  11
...

1500 rows × 7 columns
```

In [15]:

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

```
Out[15]: is_genuine      0 diagonal      0
         height_left     0 height_right    0
         margin_low      37 margin_up
         0 length        0 dtype: int64
```

In [16]:

```
df["margin_low"]=df["margin_low"].fillna(df["margin_low"].mean())
```

In [17]:

```
df["is_genuine"].unique()
```

Out[17]:

```
array([1, 0], dtype=int64)
```

In [18]:

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

Out[18]:

```
is_genuine      0 diagonal       0
height_left     0 height_right   0
margin_low      0 margin_up      0
length          0 dtype: int64
```

In [19]:

```
x=df.drop(columns=["is_genuine"])
y=df["is_genuine"]
```

In [20]:

```
x
```

Out[20]:

	diagonal	height_left	height_right	margin_low	margin_up	length
0	171.81	104.86	104.95	4.52	2.89	112.83
1	171.46	103.36	103.66	3.77	2.99	113.09
2	172.69	104.48	103.50	4.40	2.94	113.16
3	171.36	103.91	103.94	3.62	3.01	113.51
4	171.73	104.28	103.46	4.04	3.48	112.54
...	...	...	...	...	...	...
1495	171.75	104.38	104.17	4.42	3.09	111.28
1496	172.19	104.63	104.44	5.27	3.37	110.97
1497	171.80	104.01	104.12	5.51	3.36	111.95
1498	172.06	104.28	104.06	5.17	3.46	112.25
1499	171.47	104.15	103.82	4.63	3.37	112.07

1500 rows × 6 columns

In [21]:

```
y
```

Out[21]:

```
0       1 1      1
2           1
3           1
4           1
```

```
..
1495        0
1496        0
1497        0
1498        0
1499        0
Name: is_genuine, Length: 1500, dtype: int64
```

```
In [22]:  x_train,x_test,y_train,y_test,=train_test_split(x,y,train_size=0.8,random_stat
```

```
In [23]:  s=StandardScaler()
```

```
In [24]:  x_train=s.fit_transform(x_train) x_test=s.fit_transform(x_test)
```

```
In [25]:  model=LogisticRegression()
```

```
In [26]:  model.fit(x_train,y_train)
```

```
Out[26]: LogisticRegression

          LogisticRegression()
```

```
In [27]:  y_pred=model.predict(x_test)
```

```
In [28]:  y_pred
```

```
Out[28]: array([0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
                                                                                                                                 0, 1,
                0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
                1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
                0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
                1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
                0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
                0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
                1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1], dtype=int64)
```

```
In [29]:  np.array(y_test)
```

```
Out[29]: array([0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
                                                                                                                                 0, 1,
                0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
                1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
                0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
                1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
                0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
                0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
                1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,
                1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1], dtype=int64)
```

In [30]:

```python
accuracy=accuracy_score(y_test,y_pred)
```

In [31]:

```python
accuracy
```

Out[31]: 0.99

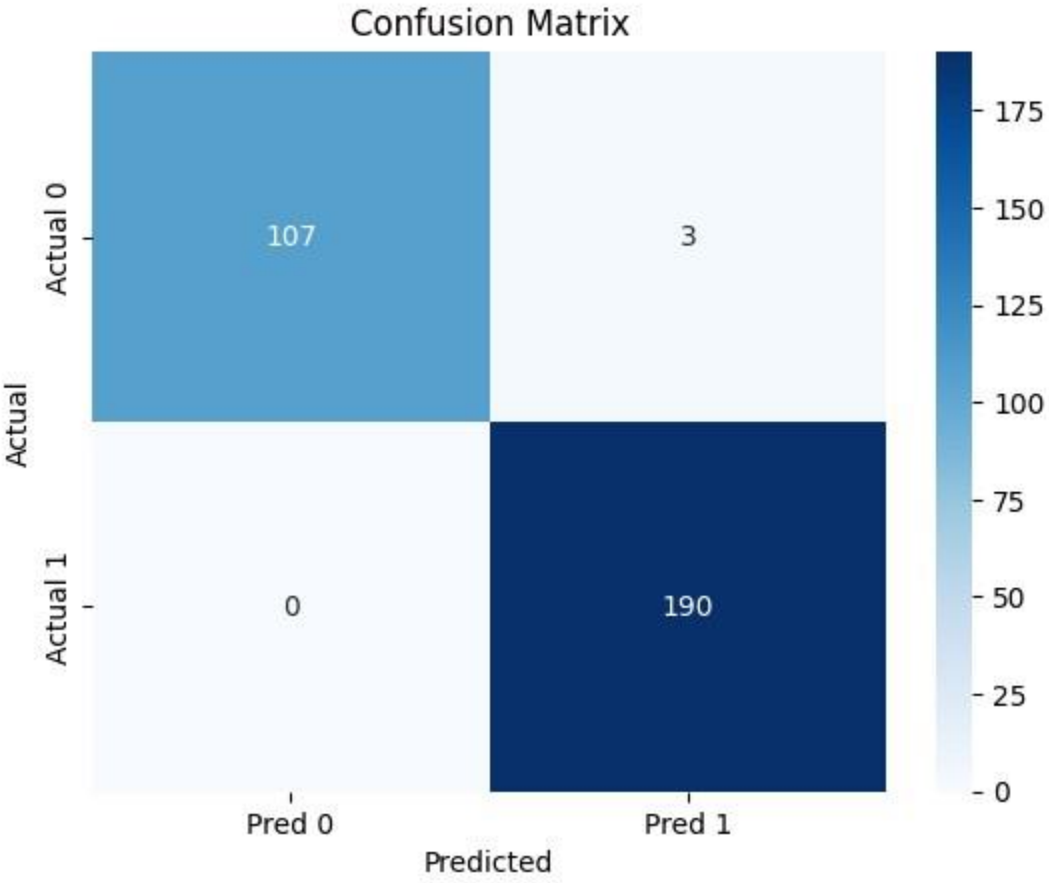
In [32]:

```python
print(classification_report(y_test,y_pred))
```

```
                                 precision    recall  f1-score   support

                            0         1.00      0.97      0.99       110
                            1         0.98      1.00      0.99       190

       accuracy                       0.99       300 macro avg       0.99      0.99      0.99
300 weighted avg       0.99      0.99      0.99       300
```

In [40]:

```python
cm = confusion_matrix(y_test, y_pred)

# Plot heatmap
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Pred 0", "Pre plt.ylabel("Actual")
plt.xlabel("Predicted") plt.title("Confusion Matrix")
plt.show()
```


Confusion Matrix
Actual
Actual 0
Actual 1
107
3
0
190
Pred 0
Pred 1
Predicted
175
150
125
100
75
50
25
0