Fake Bill Detection

In [17]:
```python
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [23]:
```python
df = pd.read_csv(r"C:\Users\Kolipaka Srikanth\AppData\Roaming\Python\Python311\Scripts\fake_bills (1).csv")
df.head()
```

Out[23]:

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	True	171.81	104.86	104.95	4.52	2.89	112.83
1	True	171.46	103.36	103.66	3.77	2.99	113.09
2	True	172.69	104.48	103.50	4.40	2.94	113.16
3	True	171.36	103.91	103.94	3.62	3.01	113.51
4	True	171.73	104.28	103.46	4.04	3.48	112.54

In [24]:
```python
df.tail()
```

Out[24]:

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
1495	False	171.75	104.38	104.17	4.42	3.09	111.28
1496	False	172.19	104.63	104.44	5.27	3.37	110.97
1497	False	171.80	104.01	104.12	5.51	3.36	111.95
1498	False	172.06	104.28	104.06	5.17	3.46	112.25
1499	False	171.47	104.15	103.82	4.63	3.37	112.07

In [25]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   is_genuine    1500 non-null   bool
 1   diagonal      1500 non-null   float64
 2   height_left   1500 non-null   float64
 3   height_right  1500 non-null   float64
 4   margin_low    1463 non-null   float64
 5   margin_up     1500 non-null   float64
 6   length        1500 non-null   float64
dtypes: bool(1), float64(6)
memory usage: 71.9 KB
```

In [26]:
```python
df.describe()
```

Out[26]:

	diagonal	height_left	height_right	margin_low	margin_up	length
count	1500.000000	1500.000000	1500.000000	1463.000000	1500.000000	1500.00000
mean	171.958440	104.029533	103.920307	4.485967	3.151473	112.67850
std	0.305195	0.299462	0.325627	0.663813	0.231813	0.87273
min	171.040000	103.140000	102.820000	2.980000	2.270000	109.49000
25%	171.750000	103.820000	103.710000	4.015000	2.990000	112.03000
50%	171.960000	104.040000	103.920000	4.310000	3.140000	112.96000
75%	172.170000	104.230000	104.150000	4.870000	3.310000	113.34000
max	173.010000	104.880000	104.950000	6.900000	3.910000	114.44000

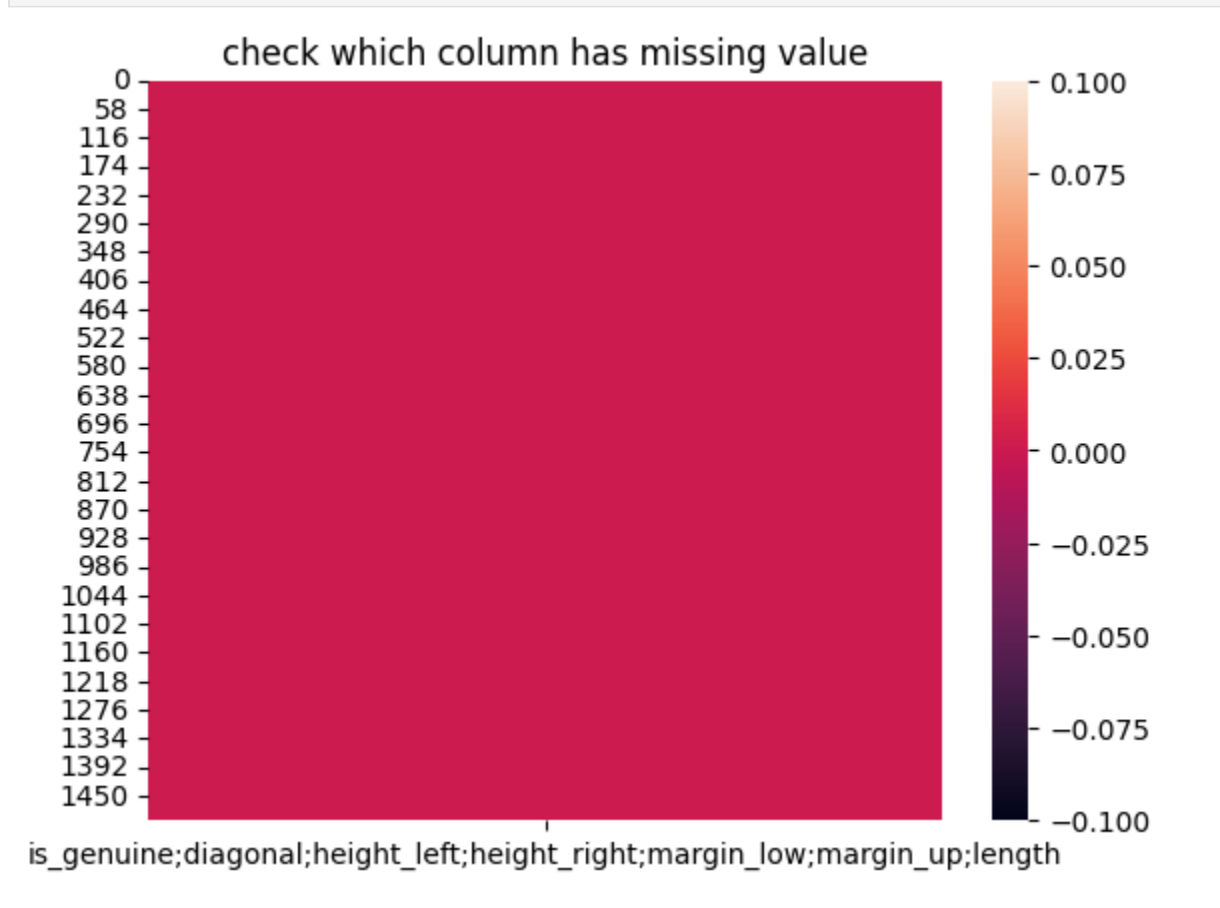
In [27]:
```python
df.isnull().sum()
```

Out[27]:
```
is_genuine       0
diagonal         0
height_left      0
height_right     0
margin_low      37
margin_up        0
length           0
dtype: int64
```

In [28]:
```python
precent_missing = df.isnull().sum() * 100 / len(df)
precent_missing
```

Out[28]:
```
is_genuine      0.000000
diagonal        0.000000
height_left     0.000000
height_right    0.000000
margin_low      2.466667
margin_up       0.000000
length          0.000000
dtype: float64
```

In [15]:
```python
sns.heatmap(df.isnull())
plt.title('check which column has missing value')
plt.show()
```



In [ ]:

In [29]:
```python
test = df[df['margin_low'].isnull()]
test['margin_low']
```

Out[29]:
```
72      NaN
99      NaN
151     NaN
197     NaN
241     NaN
251     NaN
284     NaN
334     NaN
410     NaN
413     NaN
445     NaN
481     NaN
505     NaN
611     NaN
654     NaN
675     NaN
710     NaN
739     NaN
742     NaN
780     NaN
798     NaN
844     NaN
845     NaN
871     NaN
895     NaN
919     NaN
945     NaN
946     NaN
981     NaN
1076    NaN
1121    NaN
1176    NaN
1303    NaN
1315    NaN
1347    NaN
1435    NaN
1438    NaN
Name: margin_low, dtype: float64
```

In [30]:
```python
#drop null value from dataset
train = df.dropna(inplace = True)
train
```

In [35]:
```python
#creating x-train and Y train
# x_train means 'margin-low'
x_train = df.drop('margin_low',axis = 1)

#y_train means "margiin_low"
y_train = df['margin_low']

# creating x_test
# x_test means in df['margin_low']
x_test = test.drop("margin_low",axis = 1)
```

In [38]:
```python
# linear Regression model
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

# train the model on test data
lr.fit(x_train,y_train)
```

Out[38]:
```
▾ LinearRegression
LinearRegression()
```

In [41]:
```python
# predict applying model
y_pred = lr.predict(x_test)

y_pred
```

Out[41]:
```
array([4.06495361, 4.11199026, 4.13400328, 3.99357074, 4.1403993 ,
       4.09428392, 4.07412432, 4.12538999, 4.0807278 , 4.07363322,
       4.11897255, 4.18037978, 4.13648423, 4.05106842, 4.17837685,
       4.22555104, 4.11586845, 4.10284101, 4.08184346, 4.09276238,
       4.11250192, 4.15717623, 4.16028787, 4.12193808, 4.12353555,
       4.19842271, 4.10962313, 4.09696025, 4.13384101, 5.25968515,
       5.264817  , 5.28251853, 5.30206887, 5.20035843, 5.1754678 ,
       5.17345045, 5.24675055])
```

In [42]:
```python
y_pred.shape
```

Out[42]:
```
(37,)
```

In [44]:
```python
# replace the missing value with predict value

test.loc[test.margin_low.isnull(),"margin_low"] = y_pred

test['margin_low']
```

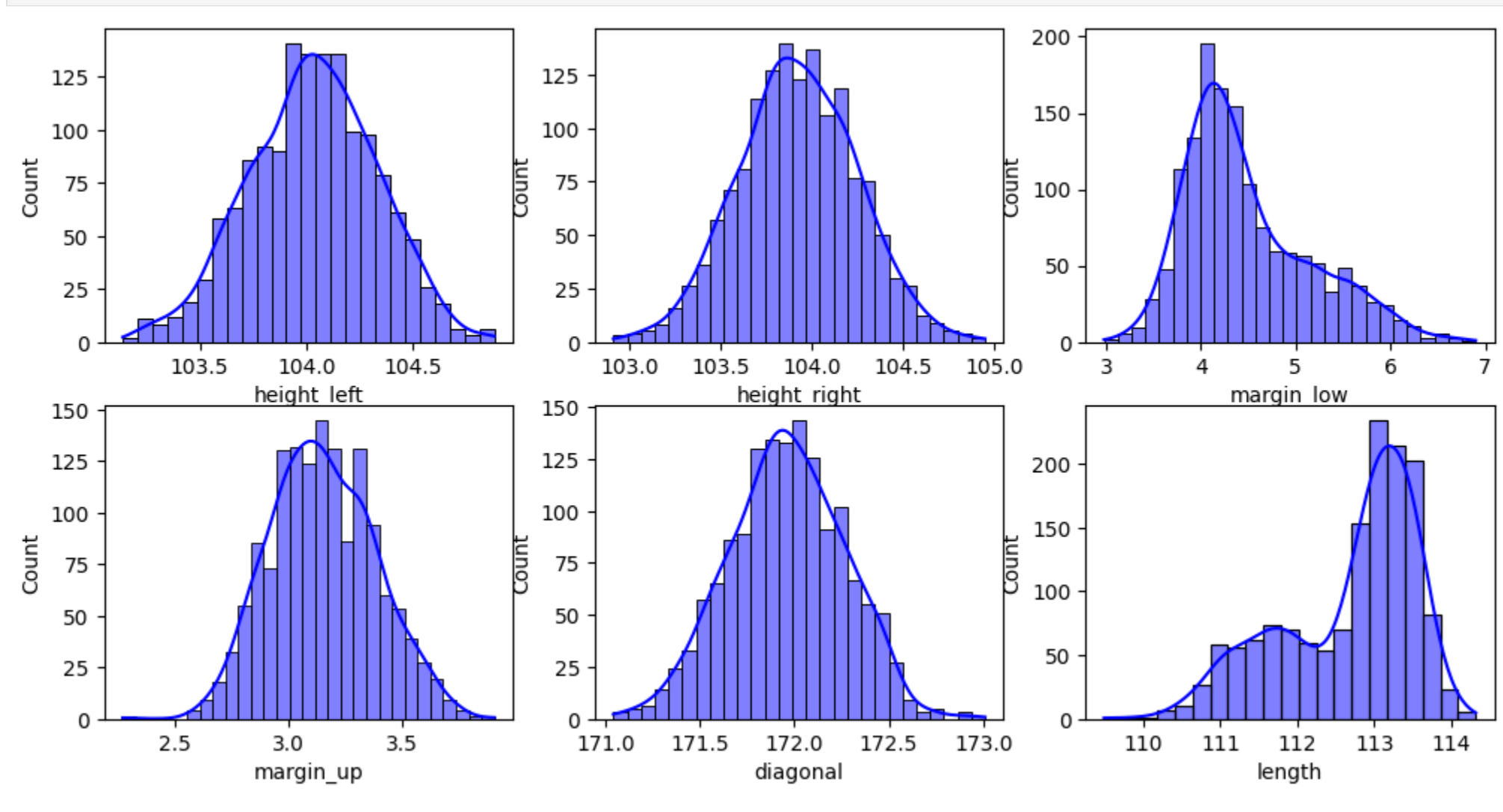
Out[44]:
```
72      4.064954
99      4.111990
151     4.134003
197     3.993571
241     4.140399
251     4.094284
284     4.074124
334     4.125390
410     4.080728
413     4.073633
445     4.118973
481     4.180380
505     4.136484
611     4.051068
654     4.178377
675     4.225551
710     4.115868
739     4.102841
742     4.081843
780     4.092762
798     4.112502
844     4.157176
845     4.160288
871     4.121938
895     4.123536
919     4.198423
945     4.109623
946     4.096960
981     4.133841
1076    5.259685
1121    5.264817
1176    5.282519
1303    5.302069
1315    5.200358
1347    5.175468
1435    5.173450
1438    5.246751
Name: margin_low, dtype: float64
```

In [46]:
```python
df.isnull().sum()
```

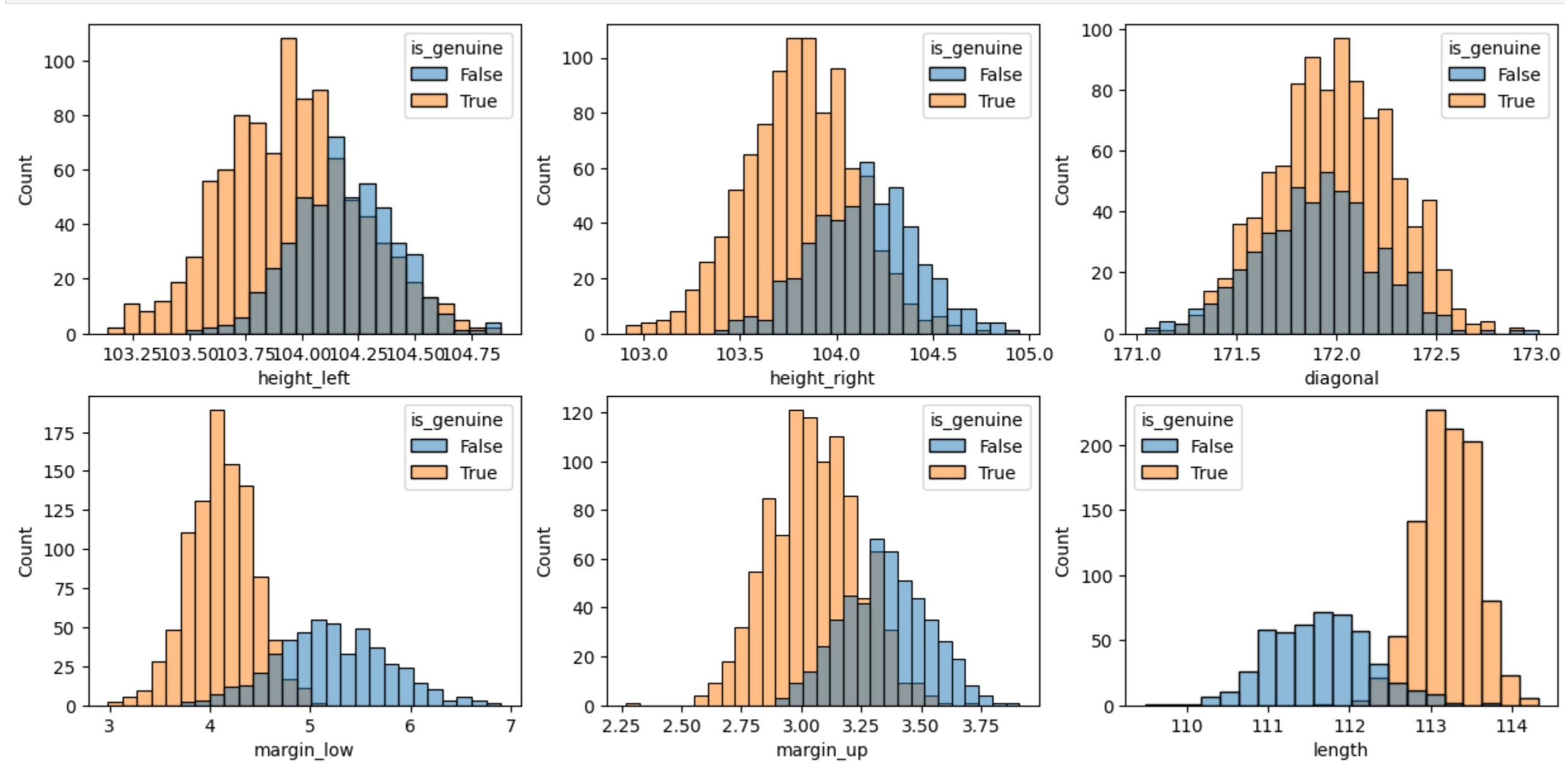
Out[46]:
```
is_genuine      0
diagonal        0
height_left     0
height_right    0
margin_low      0
margin_up       0
length          0
dtype: int64
```

Exploratory data analysis

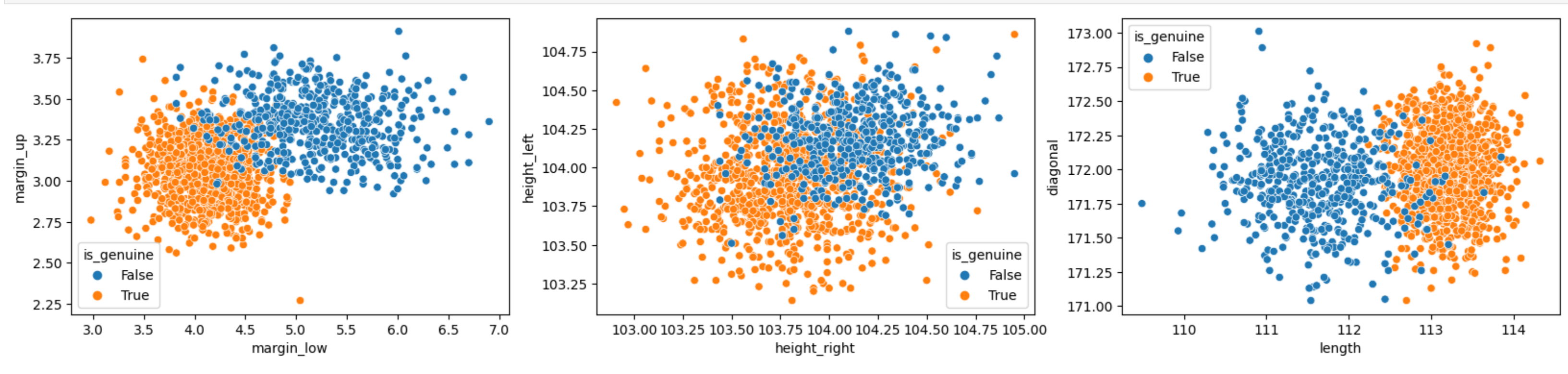
In [58]:
```python
fig,axes = plt.subplots(2,3,figsize=(12,6))
fig1 = sns.histplot(data = df,x = "height_left",kde = True,ax = axes[0,0],color='blue')
fig2 = sns.histplot(data = df,x = "height_right",kde = True,ax = axes[0,1],color='blue')
fig3 = sns.histplot(data = df,x = "margin_low",kde = True,ax = axes[0,2],color='blue')
fig4 = sns.histplot(data = df,x = "margin_up",kde = True,ax = axes[1,0],color='blue')
fig5 = sns.histplot(data = df,x = "diagonal",kde = True,ax = axes[1,1],color='blue')
fig6 = sns.histplot(data = df,x = "length",kde = True,ax = axes[1,2],color='blue')
plt.show()
```



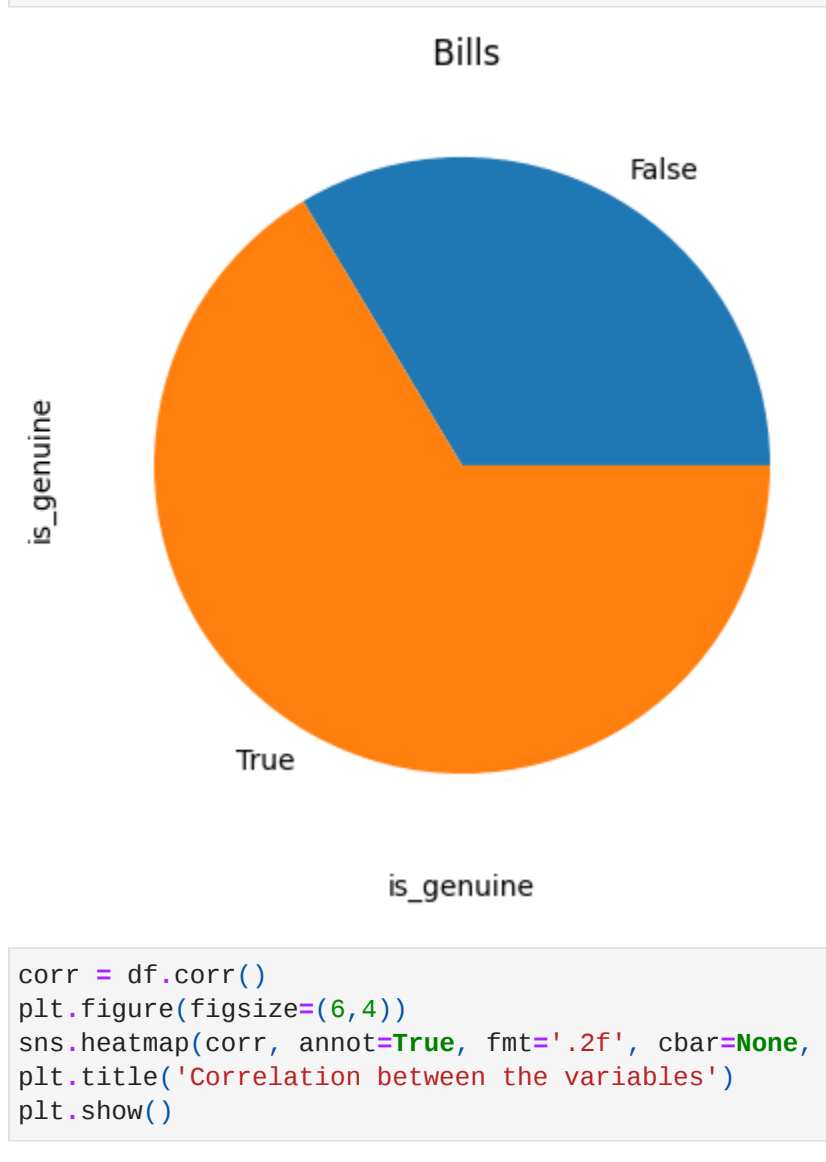
In [59]:
```python
fig,axes = plt.subplots(2,3,figsize=(15,7))
fig1 = sns.histplot(data = df,x = "height_left",hue = 'is_genuine',ax = axes[0,0])
fig2 = sns.histplot(data = df,x = "height_right",hue = 'is_genuine',ax = axes[0,1])
fig3 = sns.histplot(data = df,x = "diagonal",hue = 'is_genuine',ax = axes[0,2])
fig4 = sns.histplot(data = df,x = "margin_low",hue = 'is_genuine',ax = axes[1,0])
fig5 = sns.histplot(data = df,x = "margin_up",hue = 'is_genuine',ax = axes[1,1])
fig6 = sns.histplot(data = df,x = "length",hue = 'is_genuine',ax = axes[1,2])
plt.show()
```



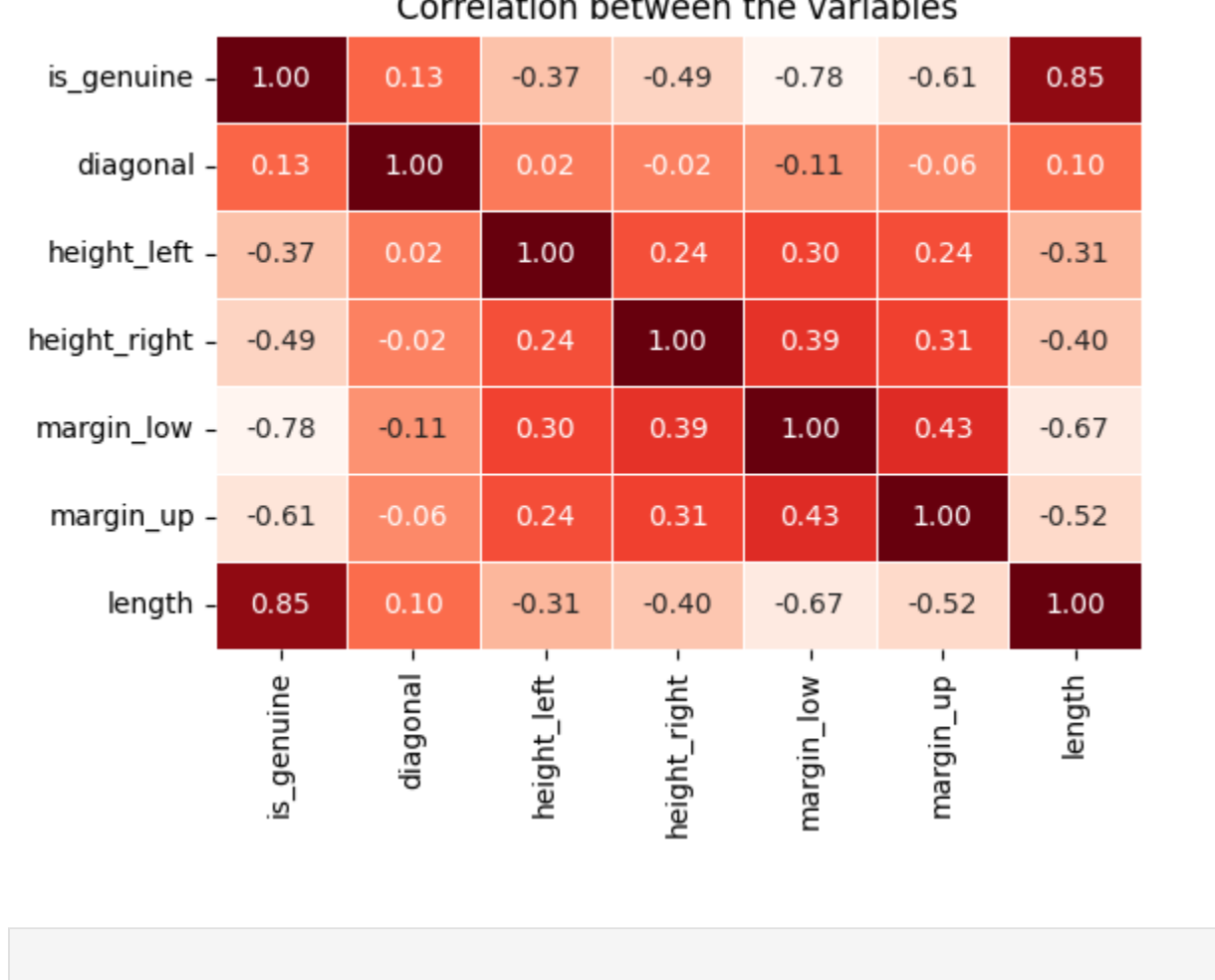
In [60]:
```python
fig,axes = plt.subplots(1,3,figsize=(20,4))
fig1 = sns.scatterplot(data = df, x = "margin_low",y = "margin_up",hue = "is_genuine",ax=axes[0])
fig1 = sns.scatterplot(data = df, x = "height_right",y = "height_left",hue = "is_genuine",ax=axes[1])
fig1 = sns.scatterplot(data = df, x = "length",y = "diagonal",hue = "is_genuine",ax=axes[2])
plt.show()
```



In [84]:
```python
plt.figure(figsize=(6,5))
plt.title(' Bills')
plt.xlabel('is_genuine')
plt.ylabel('Frequency [%]')

ax = (df.is_genuine.value_counts()/len(df)*100).sort_index().plot(kind="pie", rot=0)
```



In [83]:
```python
corr = df.corr()
plt.figure(figsize=(6,4))
sns.heatmap(corr, annot=True, fmt='.2f', cbar=None, cmap='Reds',linewidths=0.5)
plt.title('Correlation between the variables')
plt.show()
```



In [101…

In [87]:
```python
kmeans = KMeans(n_clusters=2, random_state=42)#value for k = 2
df['cluster'] = kmeans.fit_predict(df[['length', 'margin_low']])

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df[['length','margin_low','cluster']],df['is_genuine'],test_size=0.30,random_state=42)
```

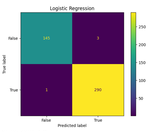
In [103…
```python
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
accuracy_logreg = accuracy_score(y_test, y_pred)

print(f'Accuracy_logreg: {accuracy_logreg} ')
```

```
Accuracy_logreg: 0.9908883826879271
```

In [104…
```python
print('Accuracy is: ', accuracy_logreg*100)
print(classification_report(y_test, y_pred))
print(metrics.confusion_matrix(y_test,y_pred))

precision_logreg = round(precision_score(y_test,y_pred,average = 'macro'),2)
recall_logreg = round(recall_score(y_test,y_pred, average = 'macro'),2)
f1_logreg = round(f1_score(y_test,y_pred, average = 'macro'),2)
accuracy_logreg = round(accuracy_score(y_test,y_pred),2)
```

```
Accuracy is:  99.08883826879271
              precision    recall  f1-score   support

       False       0.99      0.98      0.99       148
        True       0.99      1.00      0.99       291

    accuracy                           0.99       439
   macro avg       0.99      0.99      0.99       439
weighted avg       0.99      0.99      0.99       439

[[145   3]
 [  1 290]]
```

In [105…
```python
confusion_matrix = metrics.confusion_matrix(y_test,y_pred)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.title('Logistic Regression')
plt.show()
print(f'Accuracy_logreg: {accuracy_logreg}')
```




```
Accuracy_logreg: 0.99
```

In [ ]:

In [ ]: