jupyter Breast_Cancer_Prediction Last Checkpoint: 20 seconds ago

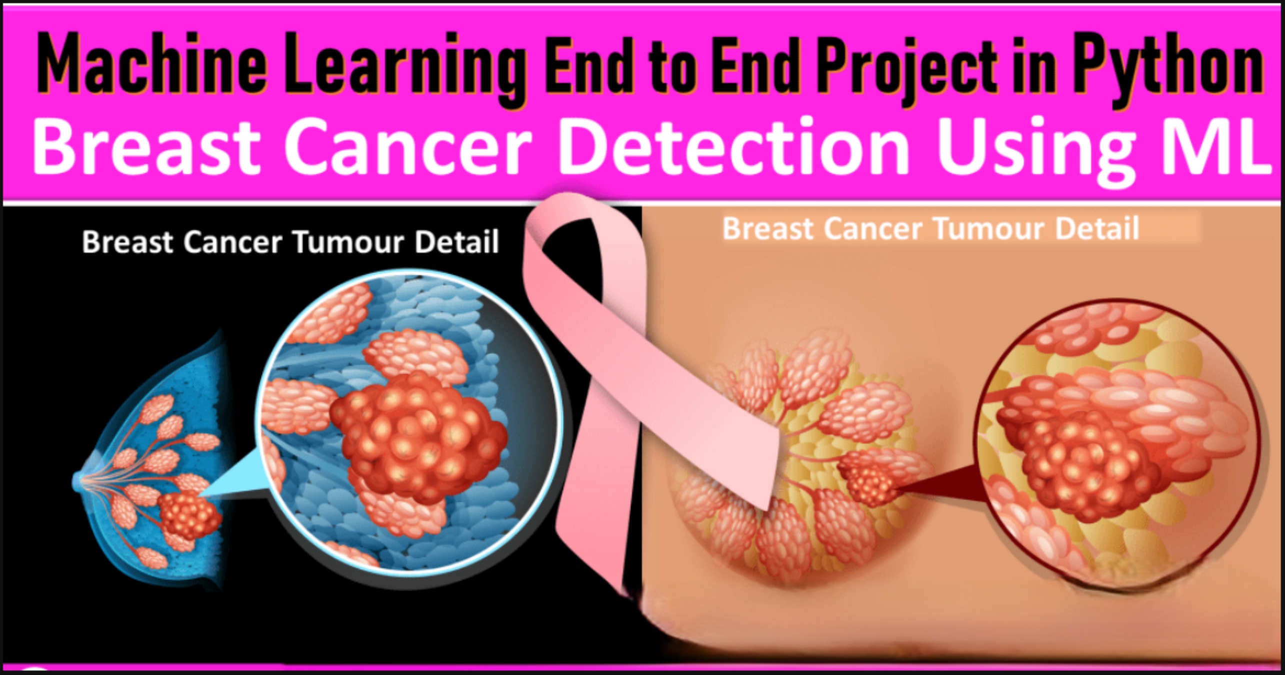
File Edit View Run Kernel Settings Help Not Trusted

Markdown JupyterLab Python 3 (ipykernel)

Author--->Abhishek Kumar

Project---> Breast Cancer Prediction Using Machine Learning Algorithm

Domain---> ***HealthCare***


Machine Learning End to End Project in Python
Breast Cancer Detection Using ML
Breast Cancer Tumour Detail
Breast Cancer Tumour Detail

```
[3]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score
```

```
[ ]:
```

```
[6]: df=pd.read_csv("breast-cancer-data.csv")
```

```
[8]: df.head()
```

[8]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	texture_worst

0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	17.33
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	23.41
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	25.53
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	26.50
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	16.67

5 rows × 33 columns

[9]:
```
#Label Encoding
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})
#Malignant
#Benign
```

[10]:
```
df.diagnosis.value_counts()
```

[10]:
```
diagnosis
0    357
1    212
Name: count, dtype: int64
```

[11]:
```
features = ['radius_mean', 'texture_mean', 'perimeter_mean', 'smoothness_mean', 'compactness_mean']

X = df[features]
y = df.diagnosis
```

[12]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2)
```

[15]:
```
len(X_train)
```

[15]:
```
455
```

[16]:
```
len(X_test)
```

[16]:
```
114
```

Model Building

[17]:
```
model = RandomForestClassifier(n_estimators=500,n_jobs=-1)
model.fit(X_train, y_train)
```

[17]:
```
▾              RandomForestClassifier
RandomForestClassifier(n_estimators=500, n_jobs=-1)
```

[18]:
```
prediction = model.predict(X_test)
```

[19]:
```
print("Accuracy is: ", round(accuracy_score(prediction, y_test)*100,2), '%')
```

```
Accuracy is:  92.98 %
```

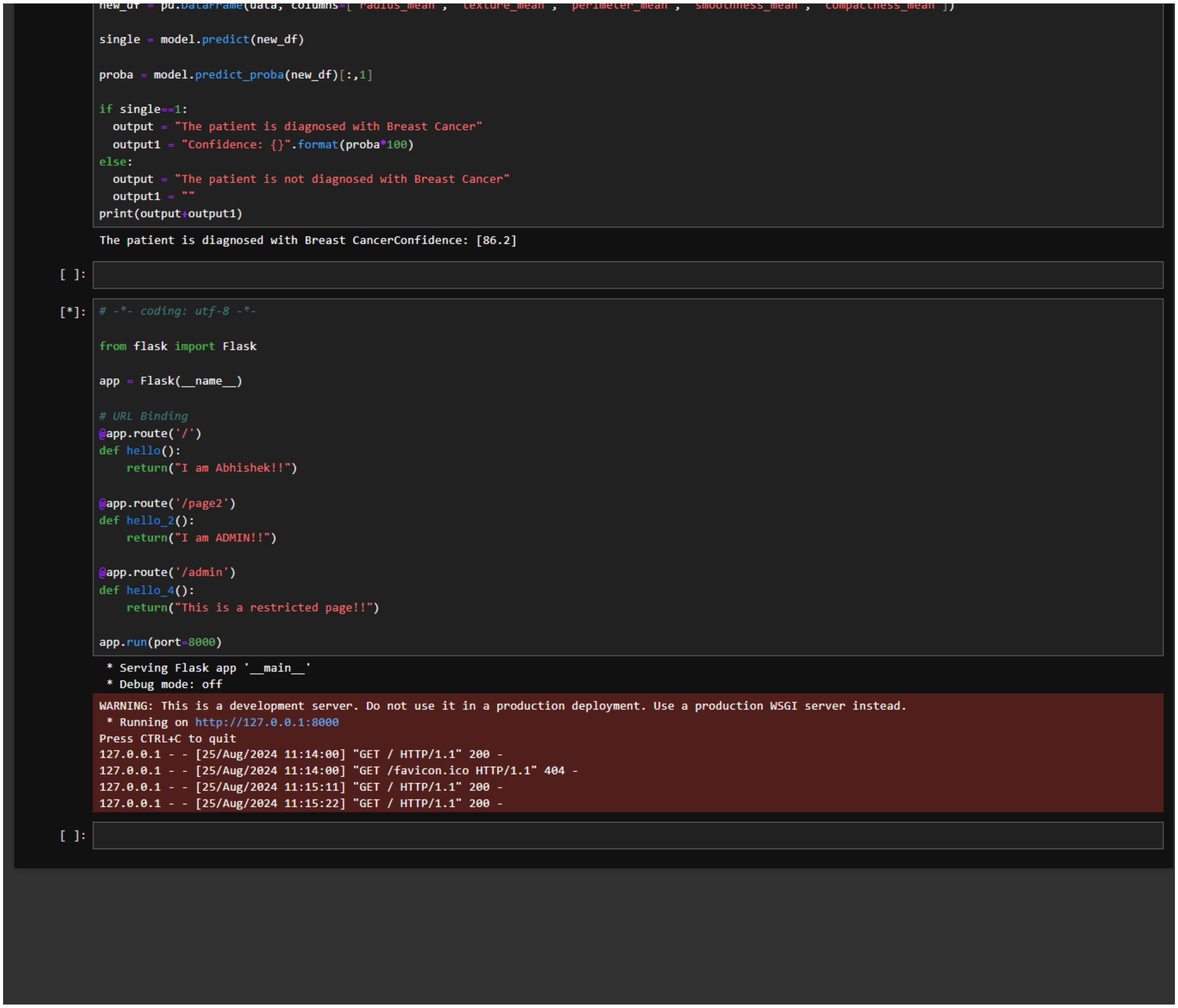
Prediction on a random datapoint

[23]:
```
import pandas as pd
```

[25]:
```
import pandas as pd

# Sample data
data = [[1, 2, 3, 4, 5]]

# Create DataFrame with appropriate column names
new_df = pd.DataFrame(data, columns=['radius_mean', 'texture_mean', 'perimeter_mean', 'smoothness_mean', 'compactness_mean'])

# Predict using the model
single = model.predict(new_df)
proba = model.predict_proba(new_df)[:, 1]

# Construct output based on prediction
if single == 1:
    output = "The patient is diagnosed with Breast Cancer"
    output1 = " Confidence: {:.2f}%".format(proba[0] * 100)
else:
    output = "The patient is not diagnosed with Breast Cancer"
    output1 = ""

# Print the results
print(output + output1)
```

```
The patient is not diagnosed with Breast Cancer
```

[26]:
```
data = [[15.99,11.38,121.8,0.1284,0.3776]]

new_df = pd.DataFrame(data, columns=['radius_mean', 'texture_mean', 'perimeter_mean', 'smoothness_mean', 'compactness_mean'])
```



```
new_df = pd.DataFrame(data, columns=['radius_mean', 'texture_mean', 'perimeter_mean', 'smoothness_mean', 'compactness_mean'])

single = model.predict(new_df)

proba = model.predict_proba(new_df)[:,1]

if single==1:
  output = "The patient is diagnosed with Breast Cancer"
  output1 = "Confidence: {}".format(proba*100)
else:
  output = "The patient is not diagnosed with Breast Cancer"
  output1 = ""
print(output+output1)
```

```
The patient is diagnosed with Breast CancerConfidence: [86.2]
```

```
[ ]:
```

```
[*]: # -*- coding: utf-8 -*-

from flask import Flask

app = Flask(__name__)

# URL Binding
@app.route('/')
def hello():
    return("I am Abhishek!!")

@app.route('/page2')
def hello_2():
    return("I am ADMIN!!")

@app.route('/admin')
def hello_4():
    return("This is a restricted page!!")

app.run(port=8000)
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:8000
Press CTRL+C to quit
127.0.0.1 - - [25/Aug/2024 11:14:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [25/Aug/2024 11:14:00] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [25/Aug/2024 11:15:11] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [25/Aug/2024 11:15:22] "GET / HTTP/1.1" 200 -
```

```
[ ]:
```