```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import  sklearn.datasets
from sklearn.model_selection import train_test_split
from sklearn  import metrics

#Load the data
from google.colab import files
upload = files.upload()
df = pd.read_csv("breast_cancer_csv.csv")
df.head(7)
```

```
<IPython.core.display.HTML object>
```

```
Saving breast_cancer_csv.csv to breast_cancer_csv (2).csv
```

{"type":"dataframe","variable_name":"df"}

```python
#Count the number of rows and columns in the dataset
df.shape
```

```
(569, 32)
```

```python
#Count of the number of empty values in each column
df.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
```

```
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

```python
#Get the count of number of rows and columns
df.shape
```

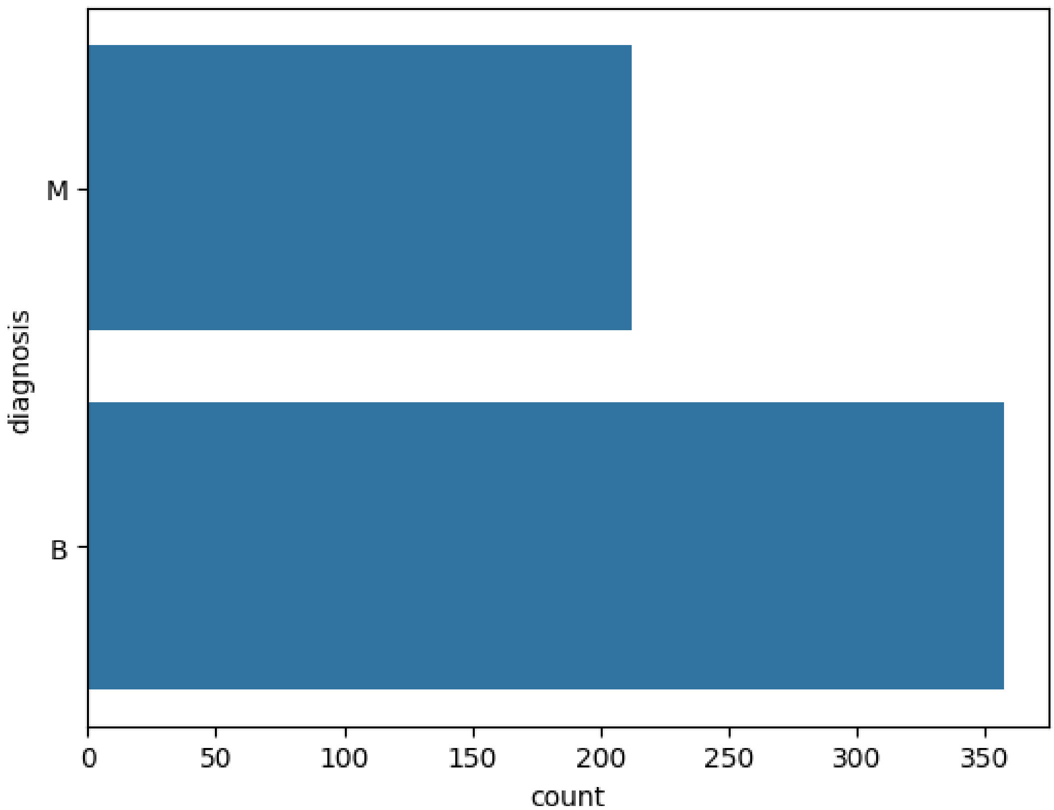
```
(569, 32)
```

```python
#Get a count of the number of Malignant (M) or Benign (B) cells
df['diagnosis'].value_counts()
```

```
diagnosis
B    357
M    212
Name: count, dtype: int64
```

```python
#Visualize the count
sns.countplot(df['diagnosis'], label ='count')
```

```
<Axes: xlabel='count', ylabel='diagnosis'>
```




```
#Look at the data types to see which columns need to be encoded
df.dtypes
```

```
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
```

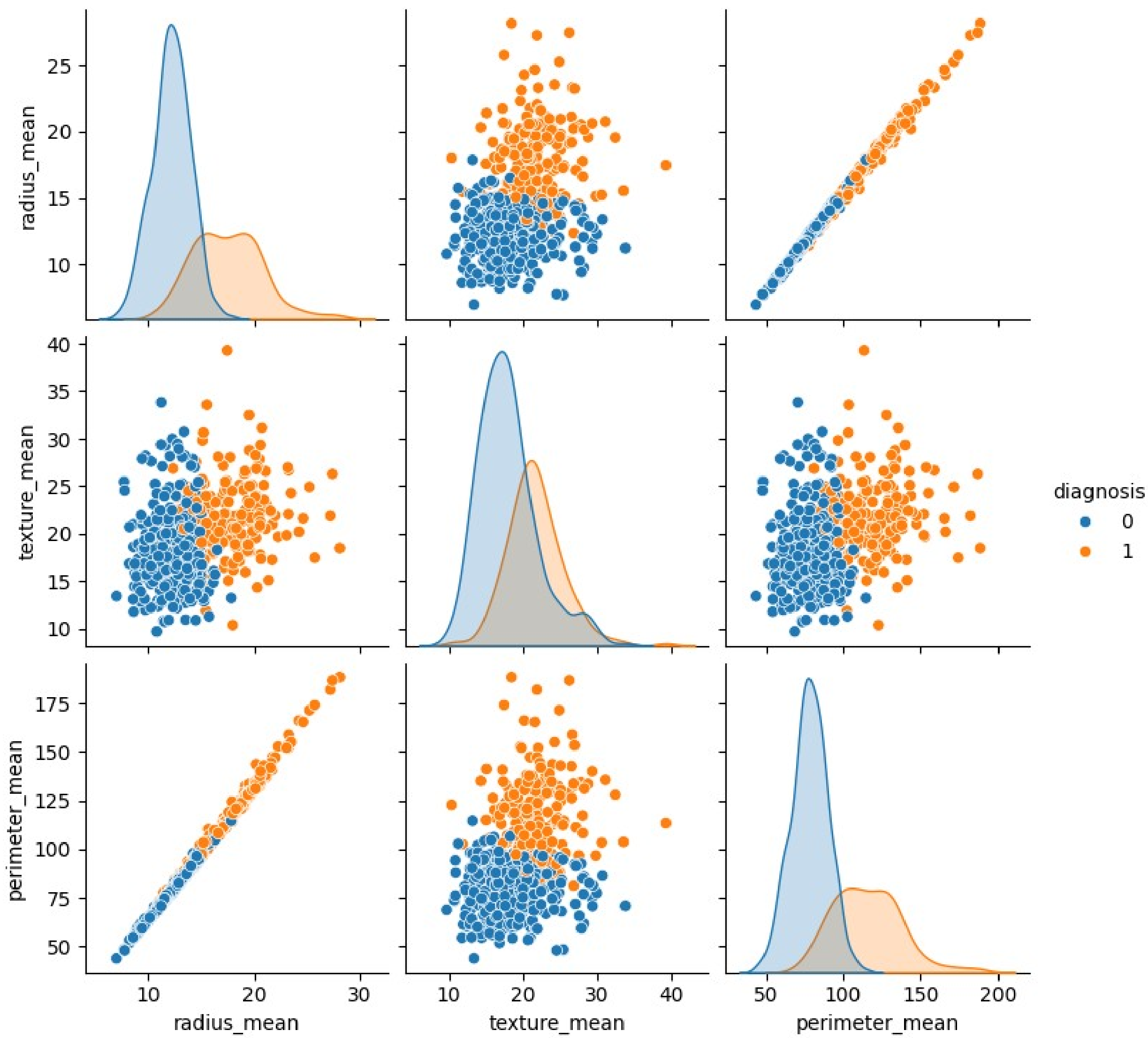
```
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
dtype: object
```

```
#Encode the categorical data Values
from sklearn.preprocessing import LabelEncoder
Labelencoder_Y = LabelEncoder()
df.iloc[:,1] = Labelencoder_Y.fit_transform(df.iloc[:,1].values)
df.iloc[:,1]
```

```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: object
```

```
#Create a pair plot
sns.pairplot(df.iloc[:,1:5],hue ='diagnosis')
```

```
<seaborn.axisgrid.PairGrid at 0x7e98d8daed10>
```




```
#Print the first five rows of the new data
df.head(5)
```

{"type":"dataframe","variable_name":"df"}

```
#Gwt the correlation of the columns
df.iloc[:,1:12].corr()
```

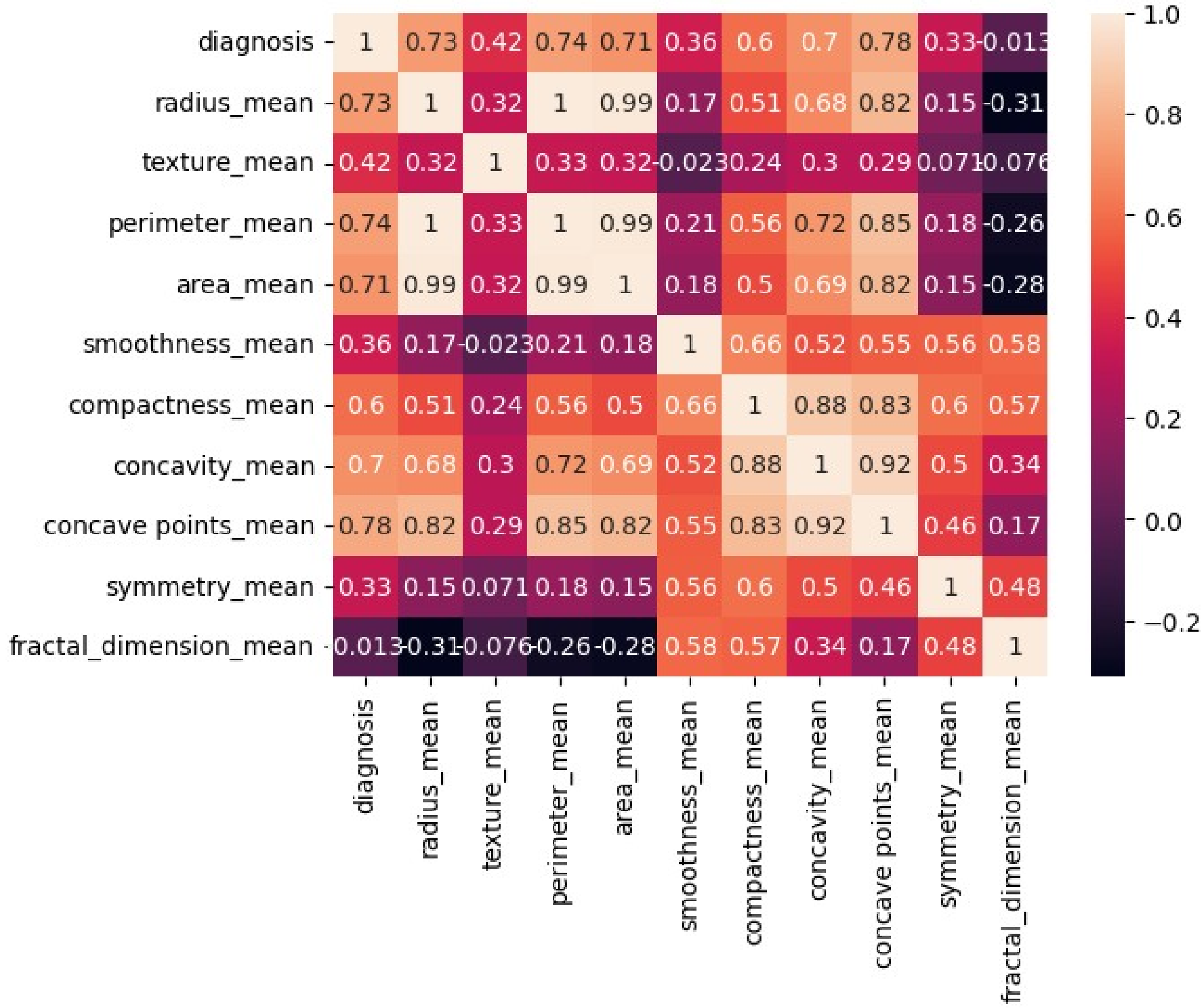
{"summary":"{\n \"name\": \"df\",\n \"rows\": 11,\n \"fields\": [\n
{\n   \"column\": \"diagnosis\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.28004378082153986,\n
\"min\": -0.012837602698431882,\n   \"max\": 1.0,\n
\"num_unique_values\": 11,\n   \"samples\": [\n
0.35855996508593324,\n   1.0,\n   0.33049855426254676\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"radius_mean\",\n

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.4264252322780478,\n        \"min\": -0.3116308263092899,\n
\"max\": 1.0,\n        \"num_unique_values\": 11,\n
\"samples\": [\n          0.17058118749299467,\n
0.7300285113754563,\n          0.14774124199260202\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"texture_mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.2840284213590684,\n        \"min\": -0.07643718344813423,\n
\"max\": 1.0,\n        \"num_unique_values\": 11,\n
\"samples\": [\n          -0.023388515998423325,\n
0.41518529984520475,\n          0.07140098048331764\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"perimeter_mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.41256121493342834,\n        \"min\": -0.26147690806633256,\n
\"max\": 1.0,\n        \"num_unique_values\": 11,\n
\"samples\": [\n          0.2072781636910072,\n
0.7426355297258334,\n          0.18302721211685316\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"area_mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.4183251195109886,\n        \"min\": -0.2831098116914261,\n
\"max\": 1.0,\n        \"num_unique_values\": 11,\n
\"samples\": [\n          0.1770283772540016,\n
0.7089838365853902,\n          0.15129307903511224\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"smoothness_mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.2885968069214828,\n        \"min\": -0.023388515998423325,\n
\"max\": 1.0,\n        \"num_unique_values\": 11,\n
\"samples\": [\n          1.0,\n          0.35855996508593324,\n
0.5577747880728878\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"compactness_mean\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.20999390796030692,\n        \"min\":
0.236702222074372,\n        \"max\": 1.0,\n
\"num_unique_values\": 11,\n        \"samples\": [\n
0.6591232152159234,\n          0.5965336775082527,\n
0.6026410484055158\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"concavity_mean\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.227129108218085,\n        \"min\":
0.30241782794389144,\n        \"max\": 1.0,\n
\"num_unique_values\": 11,\n        \"samples\": [\n
0.52198376771426,\n          0.6963597071719052,\n
0.5006666171419609\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"concave points_mean\",\n      \"properties\": {\n        \"dtype\":
```

```
\"number\",\n        \"std\": 0.2718082580983044,\n        \"min\":
0.1669173832269923,\n        \"max\": 1.0,\n
\"num_unique_values\": 11,\n        \"samples\": [\n
0.5536951727437609,\n          0.7766138400204371,\n
0.4624973883673585\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"symmetry_mean\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.2703816815678306,\n        \"min\":
0.07140098048331764,\n        \"max\": 1.0,\n
\"num_unique_values\": 11,\n        \"samples\": [\n
0.5577747880728878,\n          0.33049855426254676,\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"fractal_dimension_mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.43006351493092915,\n        \"min\": -0.3116308263092899,\n
\"max\": 1.0,\n        \"num_unique_values\": 11,\n
\"samples\": [\n          0.5847920019499775,\n          -
0.012837602698431882,\n          0.47992133005096926\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
#Visualize the correlation
sns.heatmap(df.iloc[:,1:12].corr(), annot = True)
```

```
<Axes: >
```




```
#Split the data set into independent (X) and dependent (Y) data sets
X= df.iloc[:,2:32].values
Y = df.iloc[:,1].values

#Split the data set into 75& and 25% testing
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size =
0.25, random_state = 0)

#Sccale the data [ Feature scaling]
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
X_train

array([[-0.65079907, -0.43057322, -0.68024847, ..., -0.36433881,
         0.32349851, -0.7578486 ],
       [-0.82835341,  0.15226547, -0.82773762, ..., -1.45036679,
```

```
          0.62563098, -1.03071387],
       [ 1.68277234,  2.18977235,  1.60009756, ...,  0.72504581,
        -0.51329768, -0.96601386],
       ...,
       [-1.33114223, -0.22172269, -1.3242844 , ..., -0.98806491,
        -0.69995543, -0.12266325],
       [-1.25110186, -0.24600763, -1.28700242, ..., -1.75887319,
        -1.56206114, -1.00989735],
       [-0.74662205,  1.14066273, -0.72203706, ..., -0.2860679 ,
        -1.24094654,  0.2126516 ]])
```

```
X_train.shape
```

```
(426, 30)
```

```
#Apply Machine Learning Algorithm
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
Y_train = Y_train.astype(int)
log_reg.fit(X_train, Y_train)
```

```
LogisticRegression()
```

```
Y_pred = log_reg.predict(X_test)
```

```
Y_pred
```

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
0,
       1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1])
```

```
#Checking for accuracy score for the testing data
from sklearn.metrics import accuracy_score
Y_test = Y_test.astype(int)
print(accuracy_score(Y_test, log_reg.predict(X_test)))
```

```
0.958041958041958
```

```
#Checking accuracy for the training data
from sklearn.metrics import accuracy_score
```

```
Y_train = Y_train.astype(int)
print(accuracy_score(Y_train, log_reg.predict(X_train)))
```

```
0.9906103286384976
```

New section

```
!pip install confusion_metrics
```

```
Requirement already satisfied: confusion_metrics in
/usr/local/lib/python3.11/dist-packages (0.1.0)
```

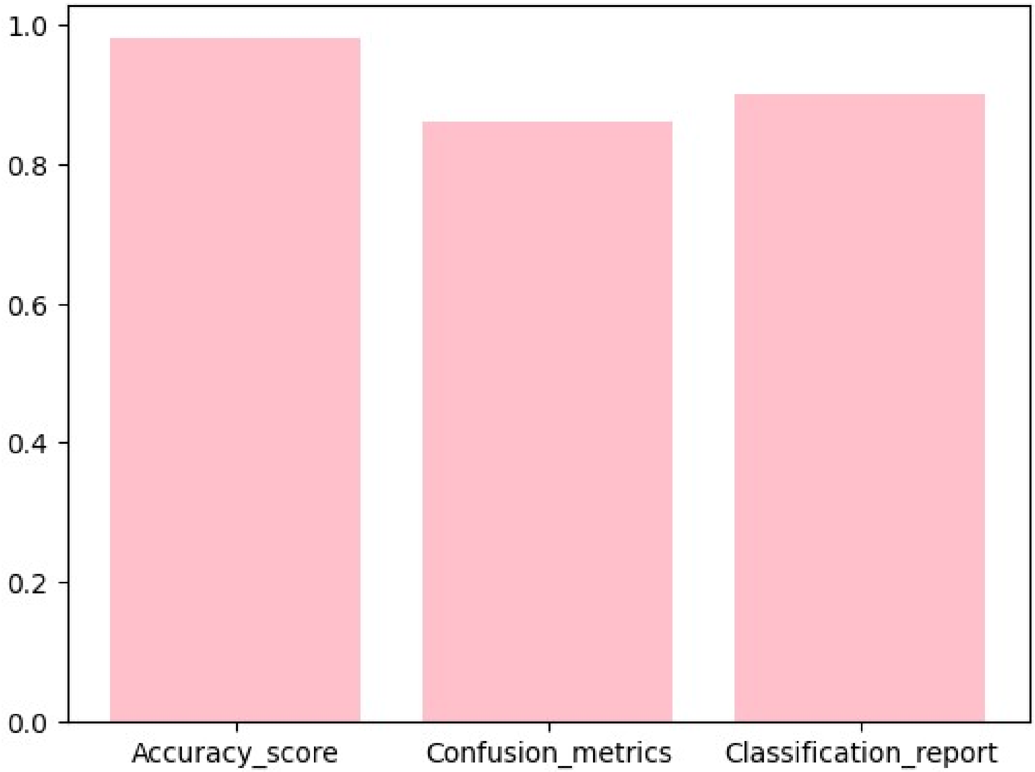
```
#Checking for confusion_metrics
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
cm
```

```
array([[86,  4],
       [ 2, 51]])
```

```
#Checking for classification report
from sklearn.metrics import classification_report
print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        90
           1       0.93      0.96      0.94        53

    accuracy                           0.96       143
   macro avg       0.95      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

```
#Visualizing data of the accuracy_score,confusion_metrics &
classification_report
import matplotlib.pyplot as plt
import seaborn as sns
plt.title = ['Diagramatic representation of accuracy_score,
confusion_metrics & classification_report']
plt.xabel = ['Display of accuracy_score, confusion_metrics &
classfication_report']
plt.ylabel = ['Logistic Regression']
x_values = ['Accuracy_score', 'Confusion_metrics',
'Classification_report']
y_values = [0.98,0.86,0.9]
plt.bar(x_values,y_values, color='pink')
plt.show()
```




```
#Using adavanced Algorithms - KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)

KNeighborsClassifier()

#Predicting data using KNN algorithm
Y_pred = knn.predict(X_test)

Y_pred

array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
0,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
0,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1])
```

```
#Checking for accuracy of testing data
from sklearn.metrics import accuracy_score
Y_test = Y_test.astype(int)
print(accuracy_score(Y_test,knn.predict(X_test)))
```

```
0.951048951048951
```

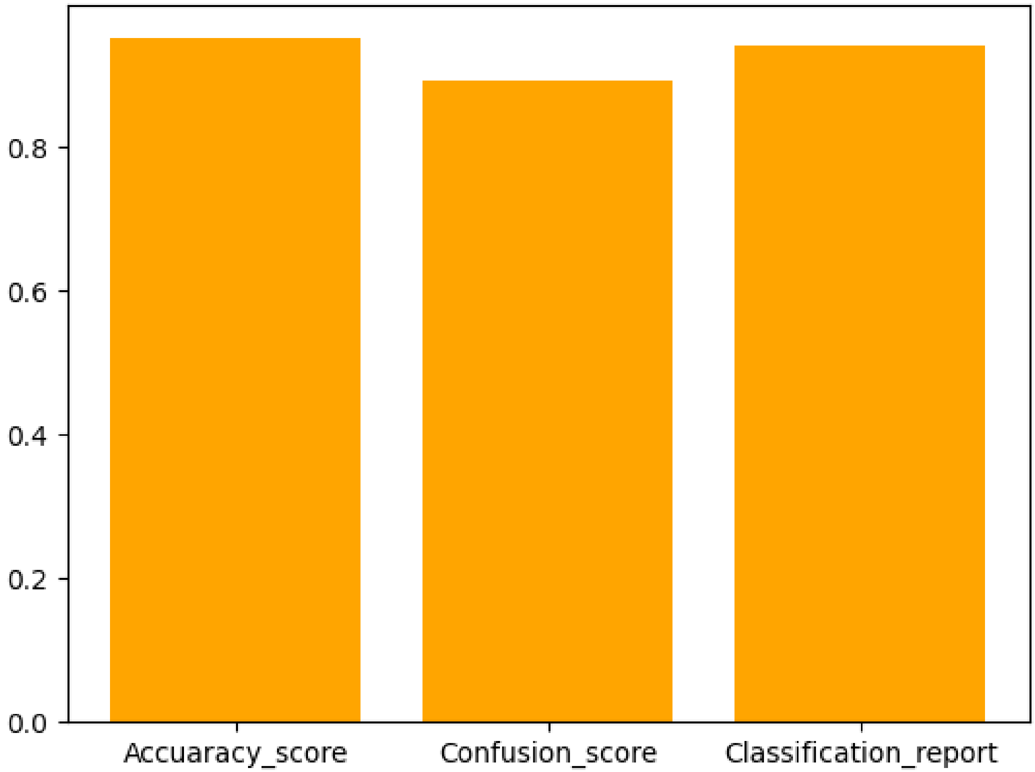
```
#Checking for confusion_metrics
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
cm
```

```
array([[89,  1],
       [ 6, 47]])
```

```
#Checking for classification report
from sklearn.metrics import classification_report
print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.99      0.96        90
           1       0.98      0.89      0.93        53

    accuracy                           0.95       143
   macro avg       0.96      0.94      0.95       143
weighted avg       0.95      0.95      0.95       143
```

```
#Visualizing the accuracy_score,confusion_metrics
classification_report
import matplotlib.pyplot as plt
import seaborn as sns
plt_set_title = ['Diagramtic representation of accuracy_score,
confusion_metrics & classification_report']
plt_set_xlabel = ['Display of accuracy_score, confusion_metrics &
classification_report']
plt_set_ylabel = ['KNN']
x_values = ['Accuaracy_score', 'Confusion_score',
'Classification_report']
y_values = [0.95,0.89,0.94]
plt.bar(x_values, y_values, color='orange')
plt.show()
```




```
#Using SVM algorithm
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, Y_train)

SVC()

#Predict data using SVM
Y_pred = svm.predict(X_test)

Y_pred

array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1,
1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
0,
       0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1,
0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
0,
       1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1])
```

```
#Checking for accuracy of testing data
from  sklearn.metrics import accuracy_score
Y_test = Y_test.astype(int)
print(accuracy_score(Y_test,svm.predict(X_test)))
```

```
0.965034965034965
```

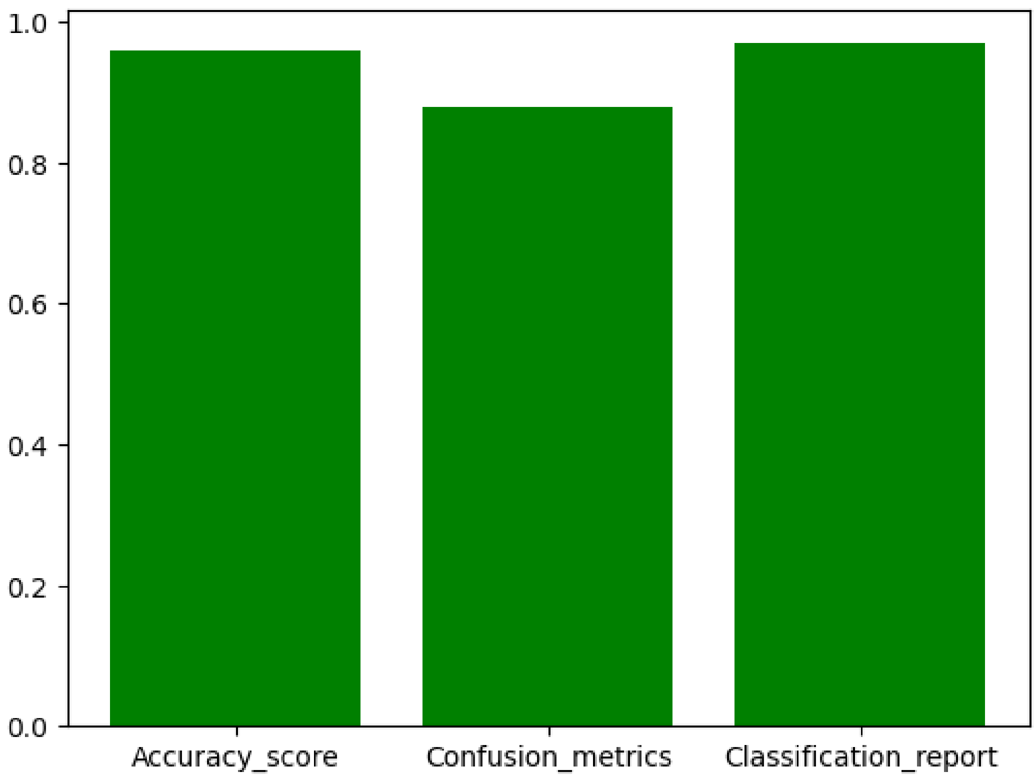
```
#Checking for confusion_metrics
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
cm
```

```
array([[88,  2],
       [ 3, 50]])
```

```
#Checking for classification report
from sklearn.metrics import classification_report
print(classification_report(Y_test, Y_pred))
```

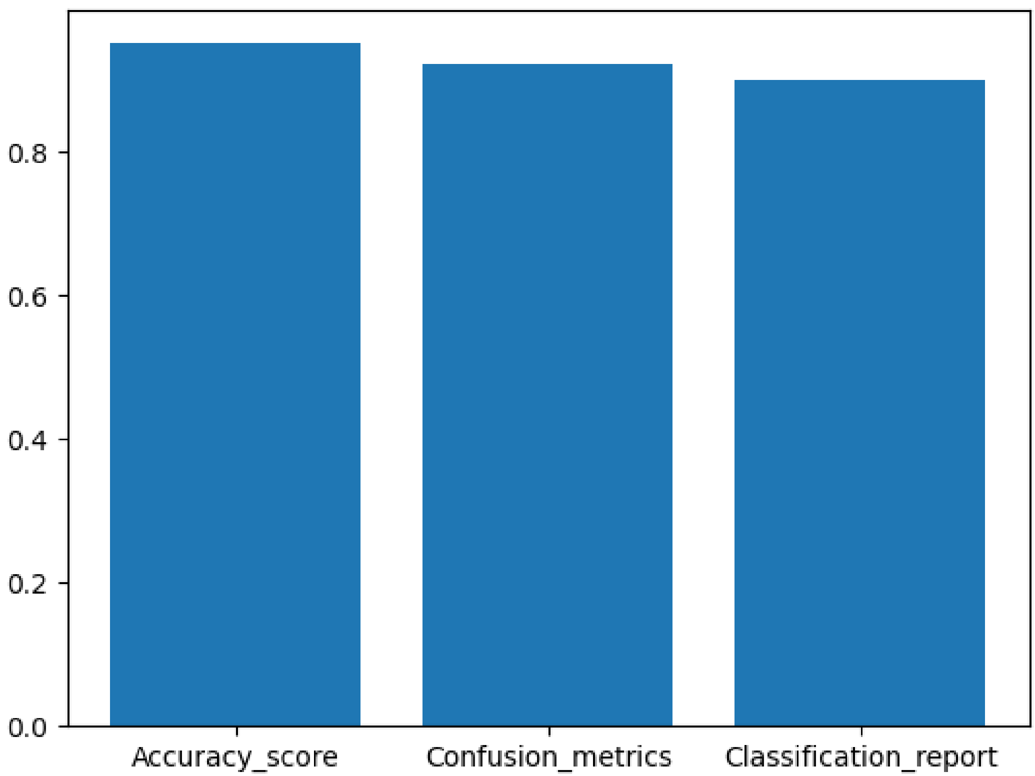
```
              precision    recall  f1-score   support

           0       0.97      0.98      0.97        90
           1       0.96      0.94      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.97      0.96       143
```

```
# Visualizing the accuracy_score, classification_score and
confusion_metrics
import matplotlib.pyplot as plt
import seaborn as sns

plt.title = ['Diagramatic representation of
accuracy_score,confusion_metrics & classification_report']
plt.xlabel = ['Dusplay accuracy_score, confusion_metrics &
classification_report']
plt.ylabel = ['SVM']
x_values = ['Accuracy_score', 'Confusion_metrics',
'Classification_report']
y_values = [0.96,0.88,0.97]
plt.bar(x_values,y_values, color='green')
plt.show()
```




```
#Using rhe Decision tree algorithm
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, Y_train)

DecisionTreeClassifier()

#Predicting data using Decsison Tree Algorithm
Y_pred = dt.predict(X_test)

Y_pred

array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1,
0,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
1,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
0,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
1,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1])
```

```
#Checking for accuracy_score
from sklearn.metrics import accuracy_score
Y_test = Y_test.astype(int)
print(accuracy_score(Y_test,dt.predict(X_test)))

0.8811188811188811

#Checking for confusion_metrics
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
cm

array([[76, 14],
       [ 3, 50]])

#Check for classificaion report
from sklearn.metrics import classification_report
Y_test = Y_test.astype(int)
print(classification_report(Y_test,Y_pred))

              precision    recall  f1-score   support

           0       0.96      0.84      0.90        90
           1       0.78      0.94      0.85        53

    accuracy                           0.88       143
   macro avg       0.87      0.89      0.88       143
weighted avg       0.90      0.88      0.88       143


#A Graph showcasing the accuracy_scorre, confusion_metrics and
classification report of thr Decision Tree
import matplotlib.pyplot as plt
import seaborn as sns
plt.set_xlabel = ['Display of accuracy_score,confusion_metrics &
classification_report']
plt.set_ylable = ['Decision Tree']
plt.set_title = ['Diagramatic representation of
accuracy_score,confusion_metrics & classification_report']
x_values =['Accuracy_score','Confusion_metrics',
'Classification_report']
y_values = [0.95,0.92,0.90]
plt.bar(x_values,y_values)
plt.show()
```


0.0
0.2
0.4
0.6
0.8
Accuracy_score
Confusion_metrics
Classification_report