NAME- RAJ KAMAL SHAKYA

LGM-VIP INTERNSHIP

BEGINNER LEVEL TASK -1

Iris Flowers Classification ML Project

1. IMPORT LIBRARY

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

1. LOAD DATA

In [ ]:
```
data=pd.read_csv("iris.csv")
```

1. UNDERSTANDING THE DATA INFO

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal length  150 non-null    float64
 1   sepal width   150 non-null    float64
 2   petal length  150 non-null    float64
 3   petal width   150 non-null    float64
 4   class         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [ ]:
```
data. describe()
```

Out[ ]:

	sepal length	sepal width	petal length	petal width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [ ]:
```
data.head()
```

Out[ ]:

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [ ]:
```
data.tail()
```

Out[ ]:

	sepal length	sepal width	petal length	petal width	class
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

1. DATA PREPROCESSING

In [ ]:
```
data=data.replace(to_replace={'class':{'Iris-setosa':0,'Iris-versicolor':1,'Iris-virginica':2}})
data.head()     #label encode
```

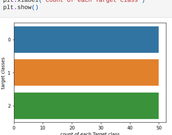
Out[ ]:

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

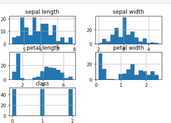
In [ ]:
```
#change data tpe of class attribut to catgoicak

data['class'] = data['class'].astype('category').cat.codes
```

In [ ]:
```
5. EXPLANTORY DATA ANALYSIS
```

In [ ]:
```
sns.countplot(y=data['class'],data=data)
plt.ylabel('target classes')
plt.xlabel('count of each Target class')
plt.show()
```

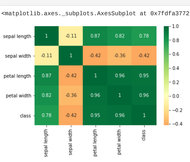


In [ ]:
```
#check the destibution of all features
data.hist(bins=15)
plt.title('Feature distibution')
plt.show()
```



In [ ]:
```
sns.heatmap(data.corr(),annot=True,cmap='RdYlGn',center=0)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdfa3772f90>
```



1. MODEL BUILDING

In [ ]:
```
# shuffling the DataFrame rows
data = data.sample(frac = 1)
data. head()
```

Out[ ]:

	sepal length	sepal width	petal length	petal width	class
58	6.6	2.9	4.6	1.3	1
0	5.1	3.5	1.4	0.2	0
116	6.5	3.0	5.5	1.8	2
99	5.7	2.8	4.1	1.3	1
71	6.1	2.8	4.0	1.3	1

In [ ]:
```
x = data.drop(['class'], axis =1)
y = data['class']
```

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 0)
```

In [ ]:
```
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

a. LOGISTIC REGRESSION

In [ ]:
```
#Creating an instance and fit the model
reg = LogisticRegression(multi_class='multinomial',solver='lbfgs')

#Fitting the train and test data
reg.fit(x_train, y_train)
```

Out[ ]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [ ]:
```
y_pred = reg.predict(x_test)
y_proba = reg.predict_proba(x_test)
y_pred
```

Out[ ]:
```
array([2, 0, 0, 2, 1, 1, 0, 1, 2, 0, 0, 2, 2, 0, 0, 1, 1, 1, 0, 1, 2, 2,
       1, 1, 1, 1, 1, 0, 2, 0, 2, 1, 1, 0, 0, 0, 2, 2, 2, 1, 2, 0, 1,
       0], dtype=int8)
```

In [ ]:
```
score_lr= reg.score(x_test,y_test)
print(score_lr)
```

```
0.9777777777777777
```

In [ ]:
```
actualy_test
predicted=y_pred
results=confusion_matrix(actual,predicted)
print('confusion matrix')
print(results)
```

```
confusion matrix
[[15  0  0]
 [ 0 15  0]
 [ 0  1 13]]
```

b. SVM

In [ ]:
```
clf = SVC(kernel='linear', C=1.0, random_state=0)
clf.fit(x_train, y_train)
```

Out[ ]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001,
    verbose=False)
```

In [ ]:
```
# generate evaluation metrics
print ("Train - Accuracy :", accuracy_score(y_train, clf.predict(x_train)))
print ("Train - Confusion matrix :\n",confusion_matrix(y_train, clf.
predict(x_train)))
```

```
Train - Accuracy : 0.9619047619047619
Train - Confusion matrix :
 [[35  0  0]
 [ 0 31  3]
 [ 0  1 35]]
```

In [ ]:
```
#print ("Train - classification report :", classification_report
#(y_train, clf.predict(x_train)))
print ("Test - Accuracy :", accuracy_score(y_test, clf.predict
(x_test)))
print ("Test - Confusion matrix :\n",confusion_matrix(y_test, clf.
predict(x_test)))
```

```
Test - Accuracy : 0.9777777777777777
Test - Confusion matrix :
 [[15  0  0]
 [ 0 16  0]
 [ 0  1 13]]
```

PREDICTION ON GIVEN INPUT

In [ ]:
```
num=[[6.4,2.9,4.3,1.3]]
num
```

Out[ ]:
```
[[6.4, 2.9, 4.3, 1.3]]
```

In [ ]:
```
num=sc.fit_transform(num)
```

In [ ]:
```
uniclf.predict(num)
u[0]
```

Out[ ]:
```
1
```

In [ ]:
```
rslt = reg.predict(num)
rslt[0]
```

Out[ ]:
```
1
```