PRESIDENCY UNIVERSITY
Presidency University Act, 2013 of the Karnataka Act No. 41 of 2013 | Established under Section 2(f) of UGC Act, 1956
Approved by AICTE, New Delhi
Itgalpur, Rajankunte, Yelahanka, Bengaluru - 560064

Name of the Department: CSE	Semester/Year:7th sem
Name of the Faculty: Ms.Poornima	Section:7IST2
Name of the Student: Poorna Chandra S	Roll No:20191IST0110
Course code:CSE235	Course Title:Introduction to Deep Learning

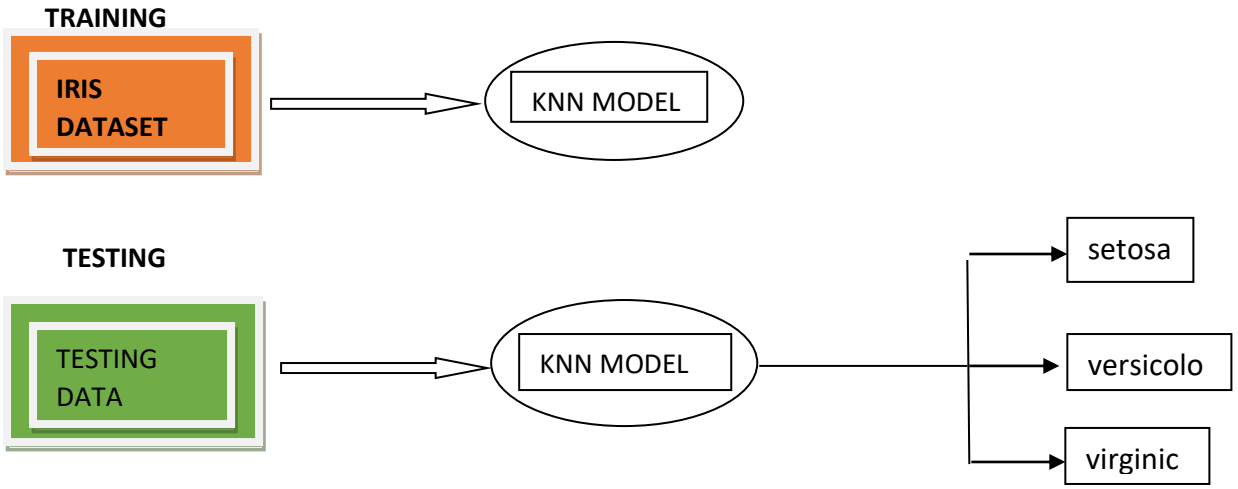
PROJECT REPORT

FLOWER SPECIES CLASSIFICATION USING KNN MODEL

Firstly we are going to dataset into two parts one is for Training and one is for Testing,

From training dataset we are going to make a KNN model i.e K Nearest Neighbour Model used for classification and once the modelling is made, we are going to test the data of the Iris(Flower) Dataset i.e we are going to take 80% of dataset for training and 20% will be taken for testing and once the modelling is made we are test the data without label,so the new model while testing will label the data into three target variables three species(setosa,versicolor,virginica)

CLASSIFICATION

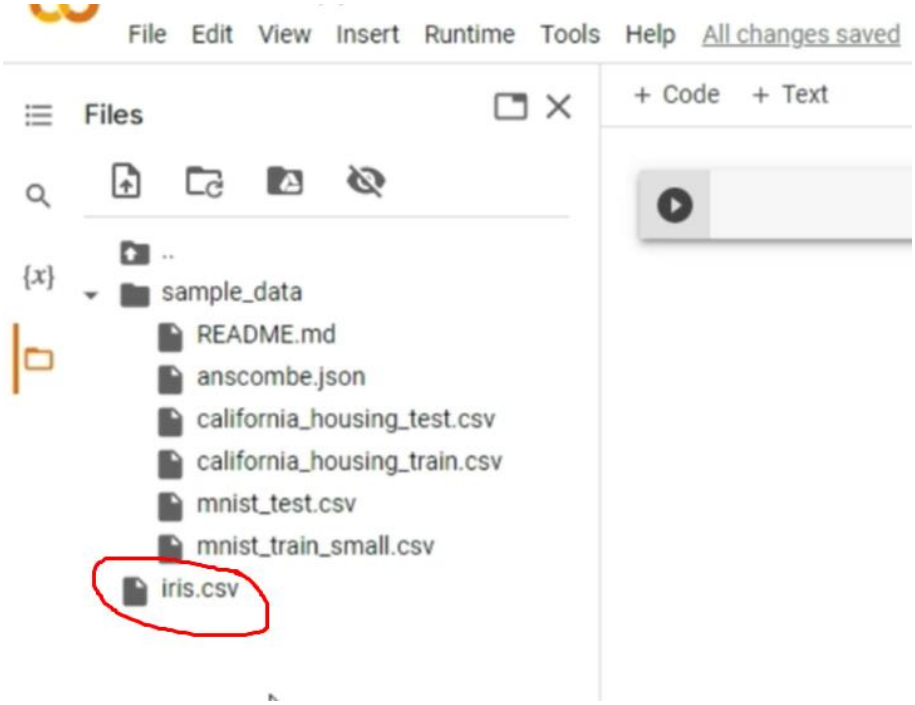


NOW TAKE A DATABASE THAT IS A CSV FILE

The databse of flower has 5 coulmns and contains features which are required for training and total features are 150 rows,here the features are sepal length ,sepal width,petal length,petal width,lastly species,in total we have 50 rows of setosa,50 rows of versicolor and 50 rows of viginica.Using this dataset Flower species classification Is done.

	A	B	C	D	E
1	sepal_len	sepal_wid	petal_len	petal_wid	species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa
11	4.9	3.1	1.5	0.1	setosa
12	5.4	3.7	1.5	0.2	setosa
13	4.8	3.4	1.6	0.2	setosa
14	4.8	3	1.4	0.1	setosa
15	4.3	3	1.1	0.1	setosa
16	5.8	4	1.2	0.2	setosa
17	5.7	4.4	1.5	0.4	setosa
18	5.4	3.9	1.3	0.4	setosa
19	5.1	3.5	1.4	0.3	setosa
20	5.7	3.8	1.7	0.3	setosa
21	5.1	3.8	1.5	0.3	setosa
22	5.4	3.4	1.7	0.2	setosa

Now upload Iris dataset to google collab



Import the required Libraries:

Importing the libraries

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn import preprocessing
    from sklearn.model_selection import train_test_split
```

We require pandas to read the dataset

Mainly numpy is used for working with Numerics like arrays

Matpltlib is used enable difficult tasks to be achieved easily with interative visualizations

Sklearn for analysis

Import KneighboursClassifier as we are using KNN

Reading the Dataset:

For reading dataset we use pd.read_csv(path of the file) after running this code the complete iris dataset will be loaded in the framework

Reading dataset

```
[ ] iris=pd.read_csv('/content/iris.csv')
```

To see the first 5 rows of iris dataset we use iris.head()

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

To see last 5 rows of dataset we use iris.tail()

```
[ ]  iris.tail()
```

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

Data analysis:

To display the number of rows and columns use .shape

Data Analysis

```
[ ]  #total rows n columns
     iris.shape
```

```
(150, 5)
```

To see count of species use iris1['species'].`value_counts()`

```
[ ]  iris['species'].value_counts()
```

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

To see the features name or column names use iris1.coumn

```
[ ]  iris.columns
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

To see the Values use `iris1.values`

```
[ ]  iris.values
```

```
array([[5.1, 3.5, 1.4, 0.2, 'setosa'],
       [4.9, 3.0, 1.4, 0.2, 'setosa'],
       [4.7, 3.2, 1.3, 0.2, 'setosa'],
       [4.6, 3.1, 1.5, 0.2, 'setosa'],
       [5.0, 3.6, 1.4, 0.2, 'setosa'],
       [5.4, 3.9, 1.7, 0.4, 'setosa'],
       [4.6, 3.4, 1.4, 0.3, 'setosa'],
       [5.0, 3.4, 1.5, 0.2, 'setosa'],
       [4.4, 2.9, 1.4, 0.2, 'setosa'],
       [4.9, 3.1, 1.5, 0.1, 'setosa'],
       [5.4, 3.7, 1.5, 0.2, 'setosa'],
       [4.8, 3.4, 1.6, 0.2, 'setosa'],
       [4.8, 3.0, 1.4, 0.1, 'setosa'],
       [4.3, 3.0, 1.1, 0.1, 'setosa'],
       [5.8, 4.0, 1.2, 0.2, 'setosa'],
       [5.7, 4.4, 1.5, 0.4, 'setosa'],
       [5.4, 3.9, 1.3, 0.4, 'setosa'],
       [5.1, 3.5, 1.4, 0.3, 'setosa'],
       [5.7, 3.8, 1.7, 0.3, 'setosa'],
       [5.1, 3.8, 1.5, 0.3, 'setosa'],
       [5.4, 3.4, 1.7, 0.2, 'setosa'],
       [5.1, 3.7, 1.5, 0.4, 'setosa'],
       [4.6, 3.6, 1.0, 0.2, 'setosa'],
       [5.1, 3.3, 1.7, 0.5, 'setosa'],
       [4.8, 3.4, 1.9, 0.2, 'setosa'],
       [5.0, 3.0, 1.6, 0.2, 'setosa'],
```

To get the info of the dataset use iris.info()

```
[ ]  # info about the dataset
     iris.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   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

To get the Statistical info of the dataset use iris.describe()

```
[ ]  #statistical info
     iris.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
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

Dataset split to x & y:

We will splitting the dataset first 4 coulmns as x which is required for training the data ,and

Target column ie. The last column to y to get the required results after training.

We use .iloc to split the data

splitting dataset

```
[ ]  x=iris.iloc[:,:4]
     y=iris.iloc[:,-1]
```

Just type x to display first 4 column and y to display last column

```
[ ]  x
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

```
[ ]  y
```

```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

Normalization of data;

We require data normalization to prepare the dataset for testing and training,the goal of normalization Is to change the value of numeric of dataset without any loss of any reading,we make sure the values of each column are closer to eachother. `preprocessing.StandardScaler().fit_transform(x)`

Data Normalization

```
[ ]  x=preprocessing.StandardScaler().fit_transform(x)
```

```
[ ]  x
```

```
       1.32509732e-01],
      [-4.16009689e-01, -1.05276654e+00,  3.64896281e-01,
       8.77547895e-04],
      [ 3.10997534e-01, -1.31979479e-01,  4.78571135e-01,
       2.64141916e-01],
      [-5.25060772e-02, -1.05276654e+00,  1.37546573e-01,
       8.77547895e-04],
      [-1.02184904e+00, -1.74335684e+00, -2.60315415e-01,
      -2.62386821e-01],
      [-2.94841818e-01, -8.22569778e-01,  2.51221427e-01,
       1.32509732e-01],
      [-1.73673948e-01, -1.31979479e-01,  2.51221427e-01,
       8.77547895e-04],
      [-1.73673948e-01, -3.62176246e-01,  2.51221427e-01,
       1.32509732e-01],
      [ 4.32165405e-01, -3.62176246e-01,  3.08058854e-01,
       1.32509732e-01],
      [-9.00681170e-01, -1.28296331e+00, -4.30827696e-01,
      -1.30754636e-01],
      [-1.73673948e-01, -5.92373012e-01,  1.94384000e-01,
       1.32509732e-01],
      [ 5.53333275e-01,  5.58610819e-01,  1.27429511e+00,
       1.71209594e+00],
      [-5.25060772e-02, -8.22569778e-01,  7.62758269e-01,
```

To train test split data use x_train,x_test, y_train,y_test

After the splitting is done .shape to check number row n coumns used for testing

Train split data

```
[ ]  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
```

```
[ ]  x_train.shape
```

```
(120, 4)
```

```
[ ]  x_test.shape
```

```
(30, 4)
```

Data Modeling and Prediction using knn model to find accuracy and classify:

Data modeling and prediction

```
[ ] knnmodel=KNeighborsClassifier(n_neighbors=3)
```

```
[ ] knnmodel.fit(x_train,y_train)

    KNeighborsClassifier(n_neighbors=3)
```

```
[ ] y_predict=knnmodel.predict(x_test)
```

Checking Accuracy

```
[ ] from sklearn.metrics import accuracy_score

    acc=accuracy_score(y_test.values,y_predict)
```

```
[ ] acc

    0.9666666666666667
```