Iris Species Data Analysis with Machine Learning

Import Libraries

In [4]:
```
import numpy as np
import random
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import plotly.offline as py
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
from plotly import tools
init_notebook_mode(connected=True)
import plotly.figure_factory as ff

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import  accuracy_score


import xgboost as xgb
import lightgbm as  lgb
from xgboost.sklearn import XGBClassifier
from catboost import CatBoostClassifier

from sklearn.preprocessing import StandardScaler, LabelBinarizer
# auxiliary function
from sklearn.preprocessing import LabelEncoder
def random_colors(number_of_colors):
    color = ["#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
                 for i in range(number_of_colors)]
    return color


import warnings
warnings.filterwarnings('ignore')
```

Load data

In [5]:
```
df = pd.read_csv('C:/Users/Iris.csv')
table = ff.create_table(df.head())
```

In [6]:
```
py.iplot(table,filename='jupyter-table1')
```

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWi
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2

In [7]:
```
py.iplot(ff.create_table(df.describe()),filename='jupyter-table1')
```

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
150.0	150.0	150.0	150.0
75.5	5.843333333333335	3.0540000000000007	3.758666666666669
43.445367992456916	0.8280661279778629	0.4335943113621737	1.7644204199522617
1.0	4.3	2.0	1.0
38.25	5.1	2.8	1.6
75.5	5.8	3.0	4.35

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

there is no null values in the data set

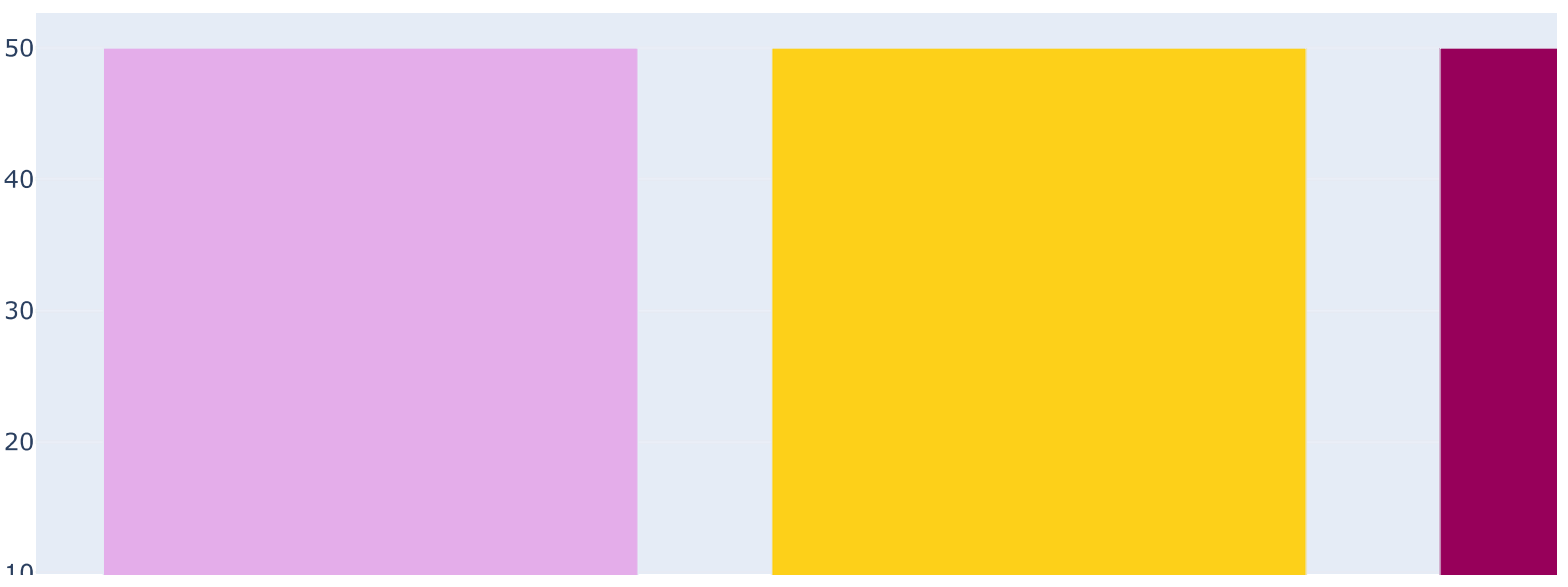
EDA

Types of Species

In [9]:
```
Species = df['Species'].unique()
Species
```

Out[9]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

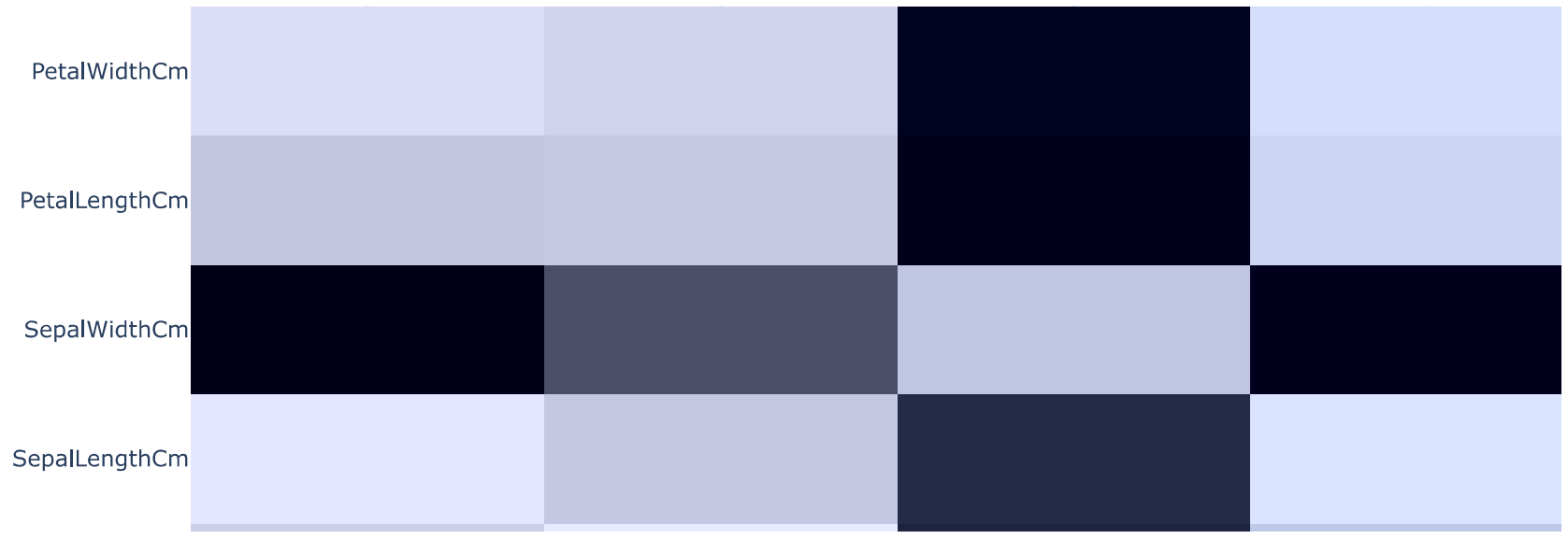
In [10]:
```
species_count = df['Species'].value_counts()
data = [go.Bar(
    x = species_count.index,
    y = species_count.values,
    marker = dict(color = random_colors(3))
)]
py.iplot(data)
```



Corelation between features

In [11]:
```
corelation = df.corr()
data = [go.Heatmap(z = np.array(corelation.values),
```

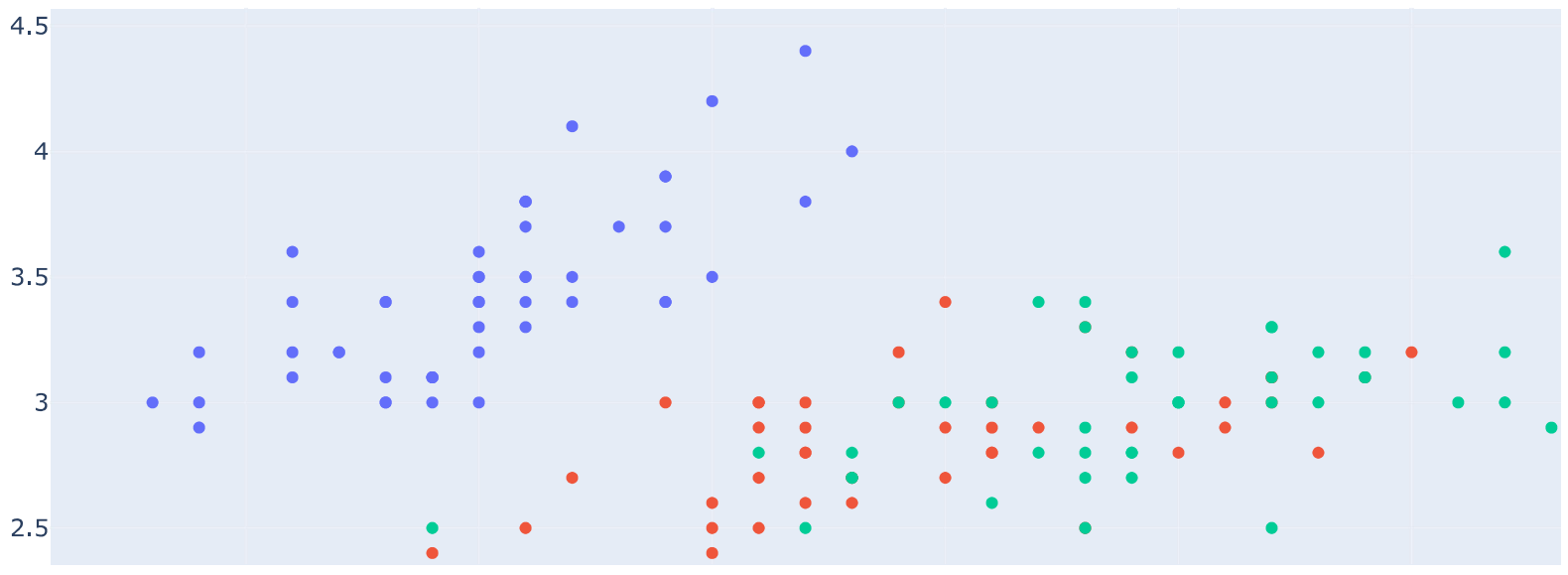
```
                    x = np.array(corelation.columns),
                    y = np.array(corelation.columns),
                      colorscale='Blackbody',)
        ]
py.iplot(data)
```



Visualizing species based on Sepal length and width

In [12]:
```
setosa = go.Scatter(x = df['SepalLengthCm'][df.Species =='Iris-setosa'], y = df['SepalWidthCm'][df.Species =='Iris-seto
                    , mode = 'markers', name = 'setosa')
versicolor = go.Scatter(x = df['SepalLengthCm'][df.Species =='Iris-versicolor'], y = df['SepalWidthCm'][df.Species =='I
                    , mode = 'markers', name = 'versicolor')
virginica = go.Scatter(x = df['SepalLengthCm'][df.Species =='Iris-virginica'], y = df['SepalWidthCm'][df.Species =='Iri
                    , mode = 'markers', name = 'virginica')
data = [setosa, versicolor, virginica]

fig = dict(data=data)
py.iplot(fig, filename='styled-scatter')
```



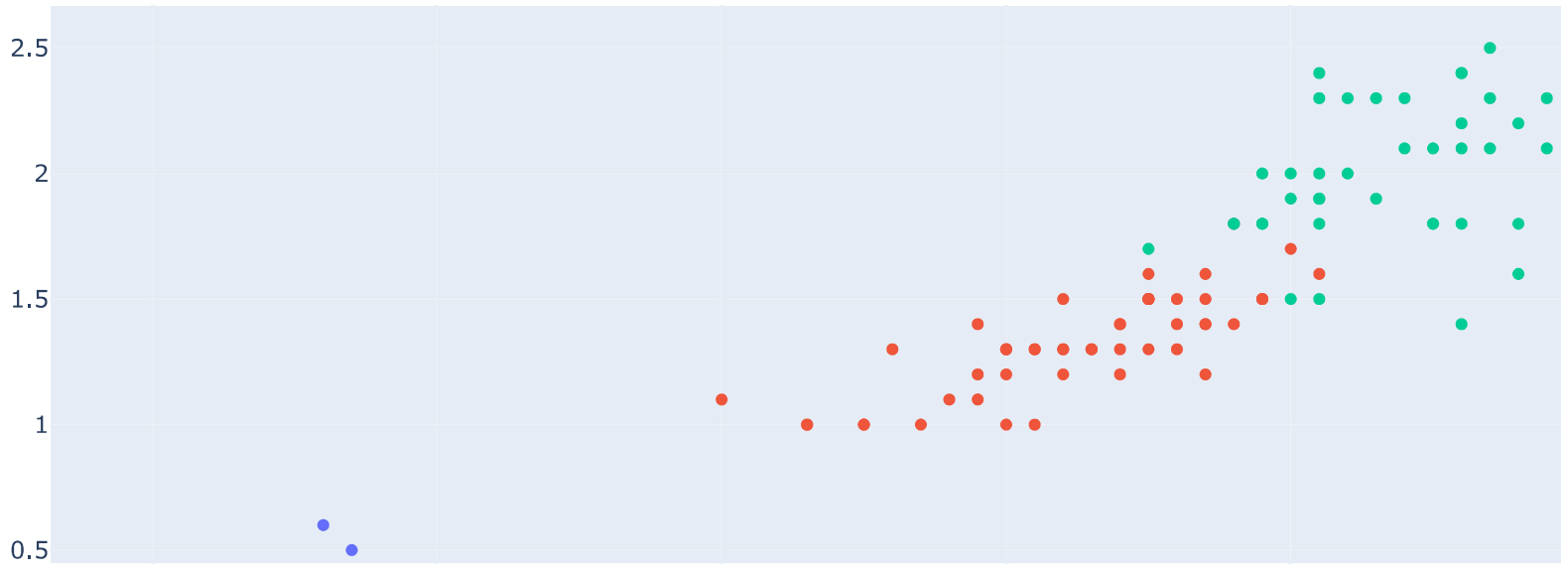
We can easily differentiate setosa based on Sepal but for versicolor and virginica its difficult because the data is scattred.

Visualizing species based on petal length and width

In [13]:
```
setosa = go.Scatter(x = df['PetalLengthCm'][df.Species =='Iris-setosa'], y = df['PetalWidthCm'][df.Species =='Iris-seto
                    , mode = 'markers', name = 'setosa')
versicolor = go.Scatter(x = df['PetalLengthCm'][df.Species =='Iris-versicolor'], y = df['PetalWidthCm'][df.Species =='I
                    , mode = 'markers', name = 'versicolor')
virginica = go.Scatter(x = df['PetalLengthCm'][df.Species =='Iris-virginica'], y = df['PetalWidthCm'][df.Species =='Iri
```

```
                    , mode = 'markers', name = 'virginica')
data = [setosa, versicolor, virginica]

fig = dict(data=data)
py.iplot(fig, filename='styled-scatter')
```



Again based on petal we can easily classify setosa and for versicolor and virginica also we can classify but there is a thin line which should be taken care of

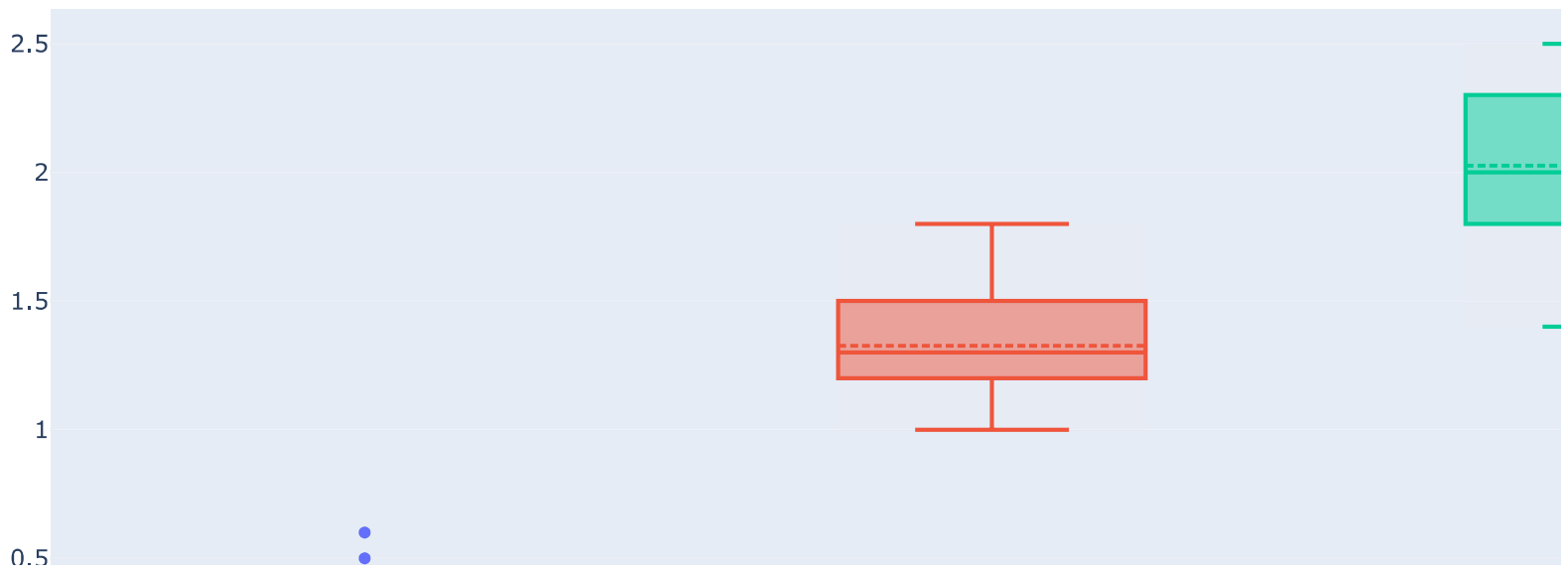
Values distribution based on petal width

In [14]:
```
trace0 = go.Box(y=df['PetalWidthCm'][df['Species'] == 'Iris-setosa'],
                boxmean=True, name = 'setosa')

trace1 = go.Box(y=df['PetalWidthCm'][df['Species'] == 'Iris-versicolor'],
                boxmean=True, name = 'versicolor')

trace2 = go.Box(y=df['PetalWidthCm'][df['Species'] == 'Iris-virginica'],
                boxmean=True, name = 'virginica')

data = [trace0, trace1, trace2]
py.iplot(data)
```



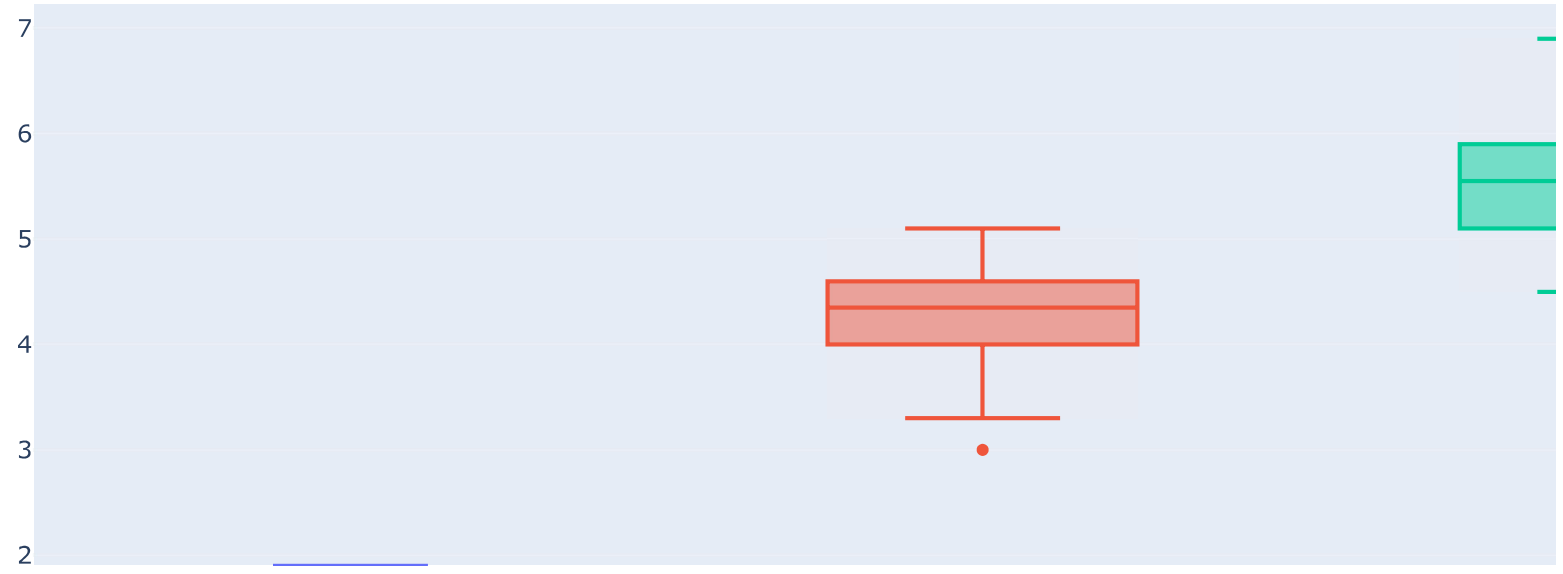
Values distribution based on petal length

In [15]:
```
trace0 = go.Box(y=df['PetalLengthCm'][df['Species'] == 'Iris-setosa'],name = 'setosa')

trace1 = go.Box(y=df['PetalLengthCm'][df['Species'] == 'Iris-versicolor'], name = 'versicolor')
```

```
trace2 = go.Box(y=df['PetalLengthCm'][df['Species'] == 'Iris-virginica'], name = 'virginica')

data = [trace0, trace1, trace2]
py.iplot(data)
```



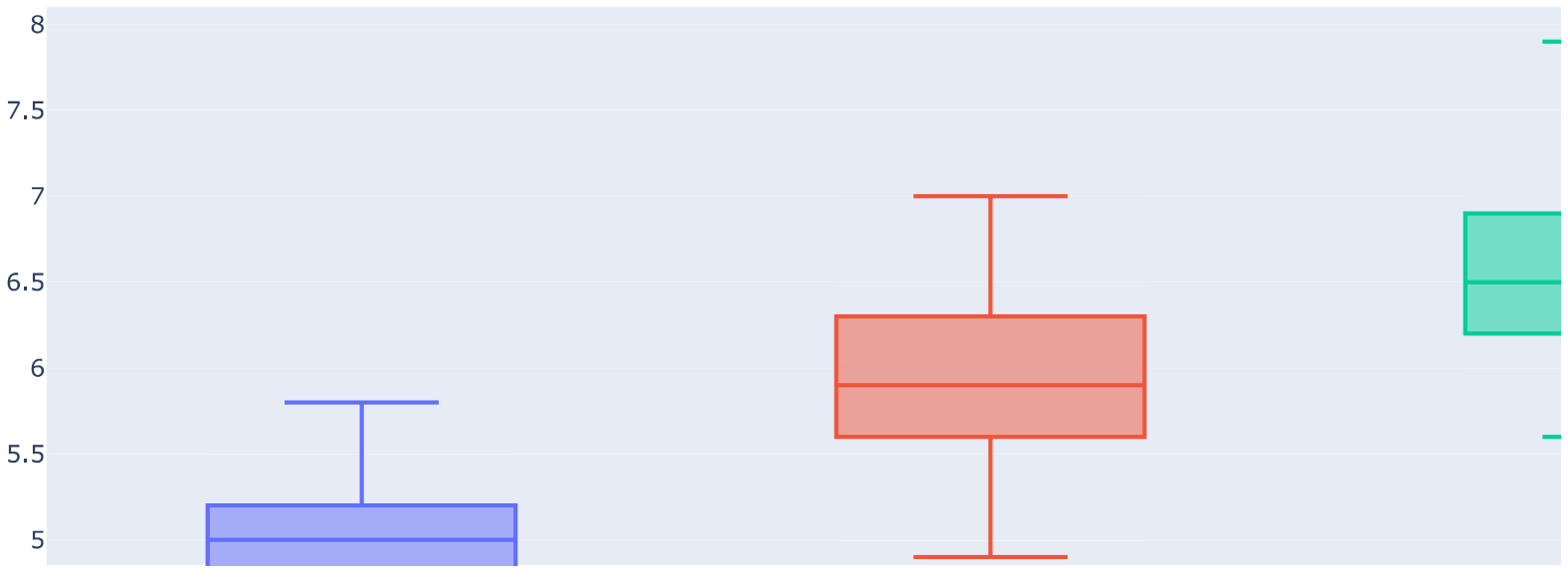
Values distribution based on sepal length

In [16]:
```
trace0 = go.Box(y=df['SepalLengthCm'][df['Species'] == 'Iris-setosa'], name = 'setosa')

trace1 = go.Box(y=df['SepalLengthCm'][df['Species'] == 'Iris-versicolor'], name = 'versicolor')

trace2 = go.Box(y=df['SepalLengthCm'][df['Species'] == 'Iris-virginica'], name = 'virginica')

data = [trace0, trace1, trace2]
py.iplot(data)
```



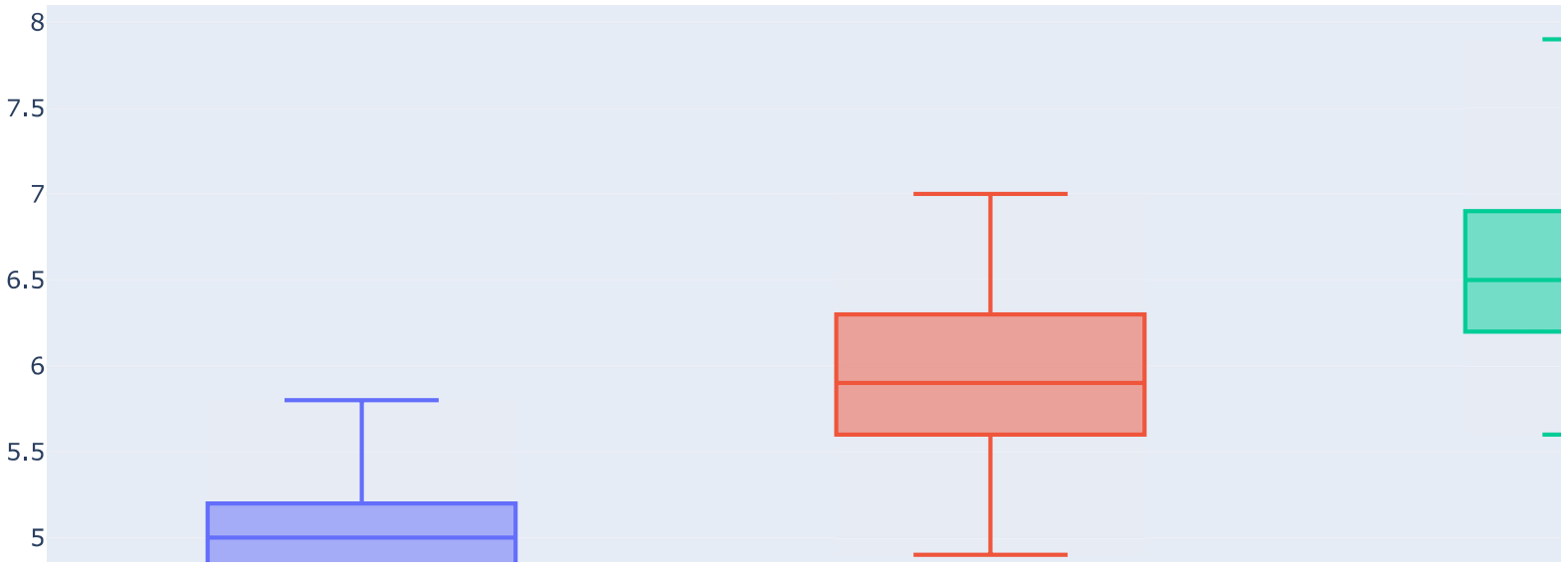
Values distribution based on sepal width

In [17]:
```
setosa = go.Box(y=df['SepalWidthCm'][df['Species'] == 'Iris-setosa'])

versicolor = go.Box(y=df['SepalWidthCm'][df['Species'] == 'Iris-versicolor'])

virginica = go.Box(y=df['SepalWidthCm'][df['Species'] == 'Iris-virginica'])

data = [trace0, trace1, trace2]
py.iplot(data)
```

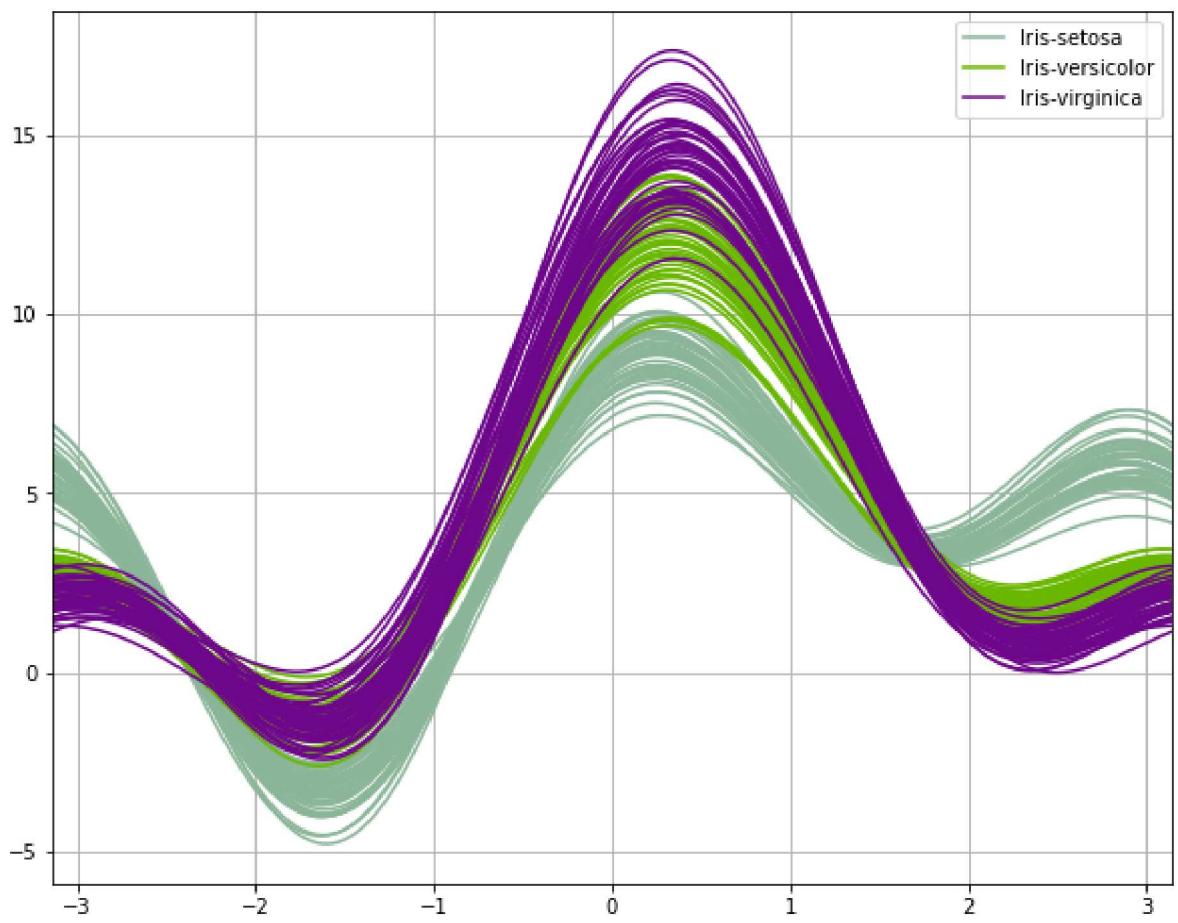


- From the above four graph you can see that the distribution of setosa < vericolor < virginica
- There are few outliers which can be explained by the scatter plot graph.

Andrew curves

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt


# If you want to create Andrews Curves:
plt.figure(figsize=(10, 8))
pd.plotting.andrews_curves(df.drop("Id", axis=1), "Species")
plt.show()
```



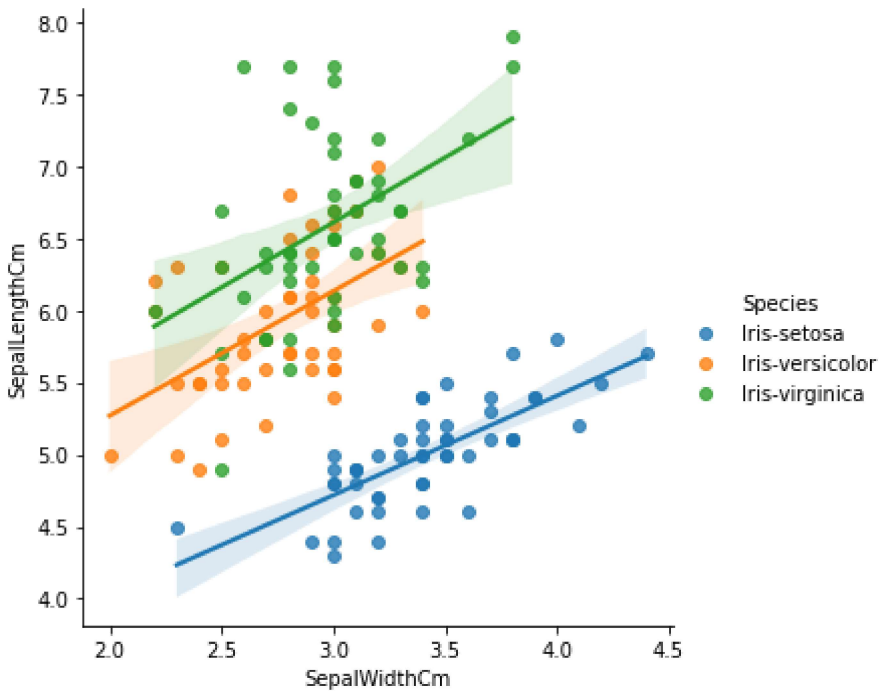
Andrews curves are a method for visualizing multidimensional data by mapping each observation onto a function.

Source - https://dzone.com/articles/andrews-curves

Lets create a regression plot for both petal and sepal

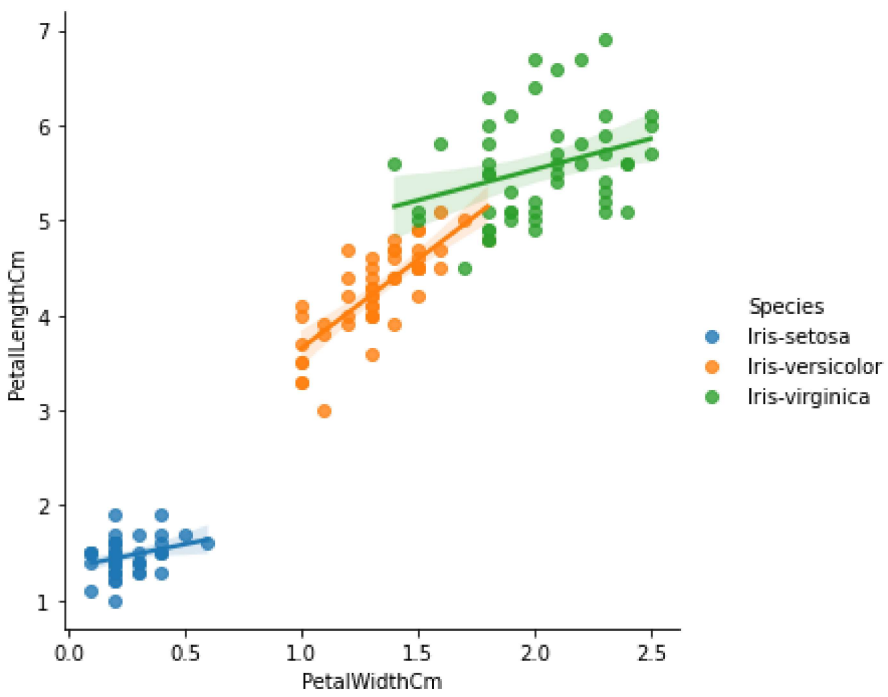
Linear regression based on sepal

In [21]:
```
g = sns.lmplot(x="SepalWidthCm", y="SepalLengthCm", hue="Species", data=df)
```



Linear regression based on petal

In [22]:
```
g = sns.lmplot(x="PetalWidthCm", y="PetalLengthCm", hue="Species", data=df)
```



Machine Learning Models

In [23]:
```
x = df[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]
y = df['Species']
```

In [24]:
```
encoder = LabelEncoder()
y = encoder.fit_transform(y)
```

In [25]:
```
y
```

Out[25]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

As you can see Iris-setosa Iris-versicolor Iris-virginica are converted to 0, 1, 2 respectively

First we are splitting the data set into training data and testing data which is 7:3 ratio

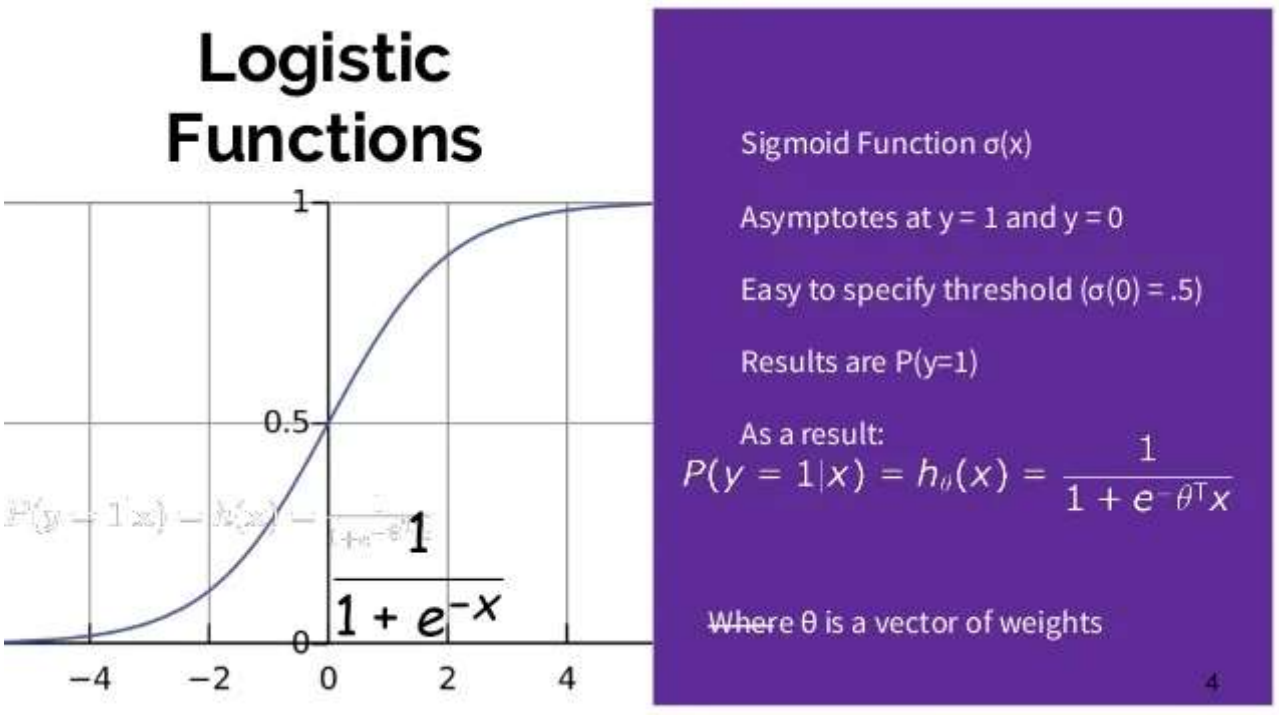
In [26]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 101)
```

List of Machine learning algorithms

Since it is a classification problem we will be using

Logistic regression
Decision tree
KNN
SVM
Naive Bayes Classification
Random forest
XGBoost
LightGBM

Logistic regression



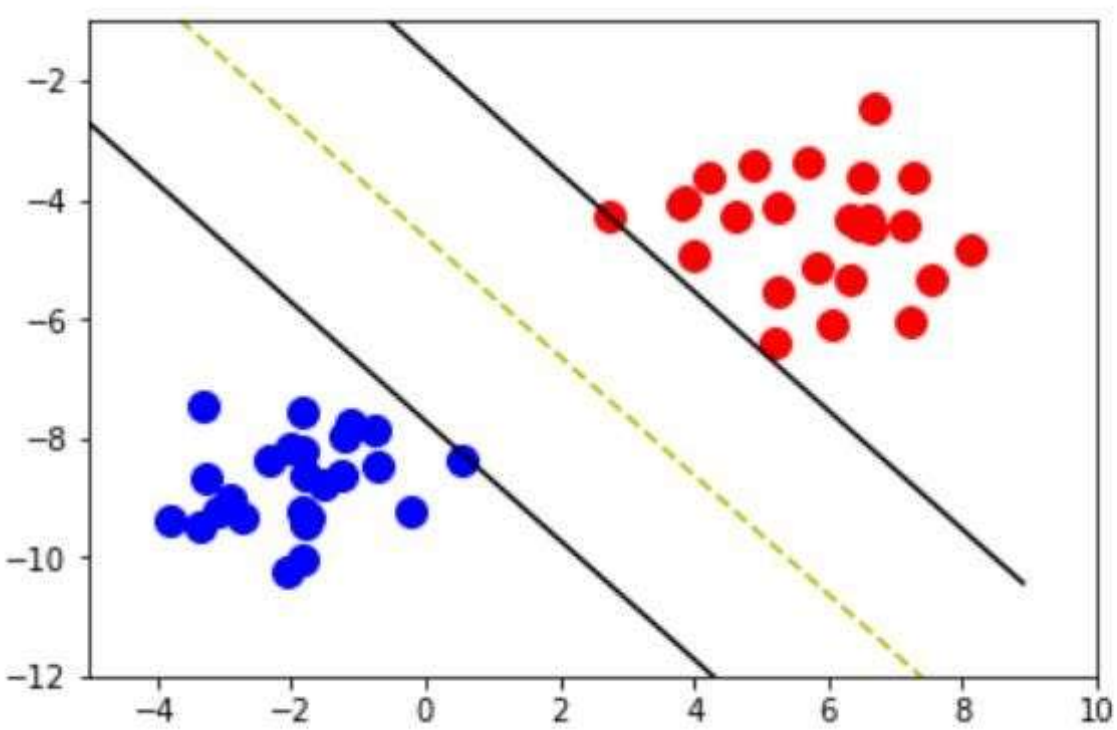
Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

In [27]:
```
lr_model = LogisticRegression()
lr_model.fit(x_train,y_train)
lr_predict = lr_model.predict(x_test)

print('Logistic Regression - ',accuracy_score(lr_predict,y_test))
```

```
Logistic Regression -  0.9777777777777777
```

SVM



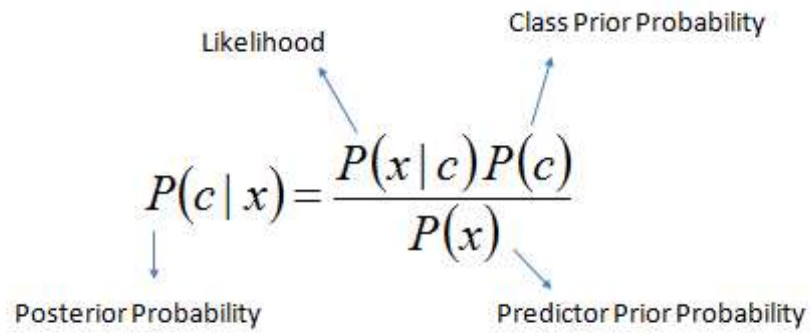
"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.

In [28]:
```
svm_model = SVC(kernel='linear')
svm_model.fit(x_train,y_train)
svc_predict = svm_model.predict(x_test)

print('SVM - ',accuracy_score(svc_predict,y_test))
```

```
SVM -  1.0
```

Naive Bayes Classification



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

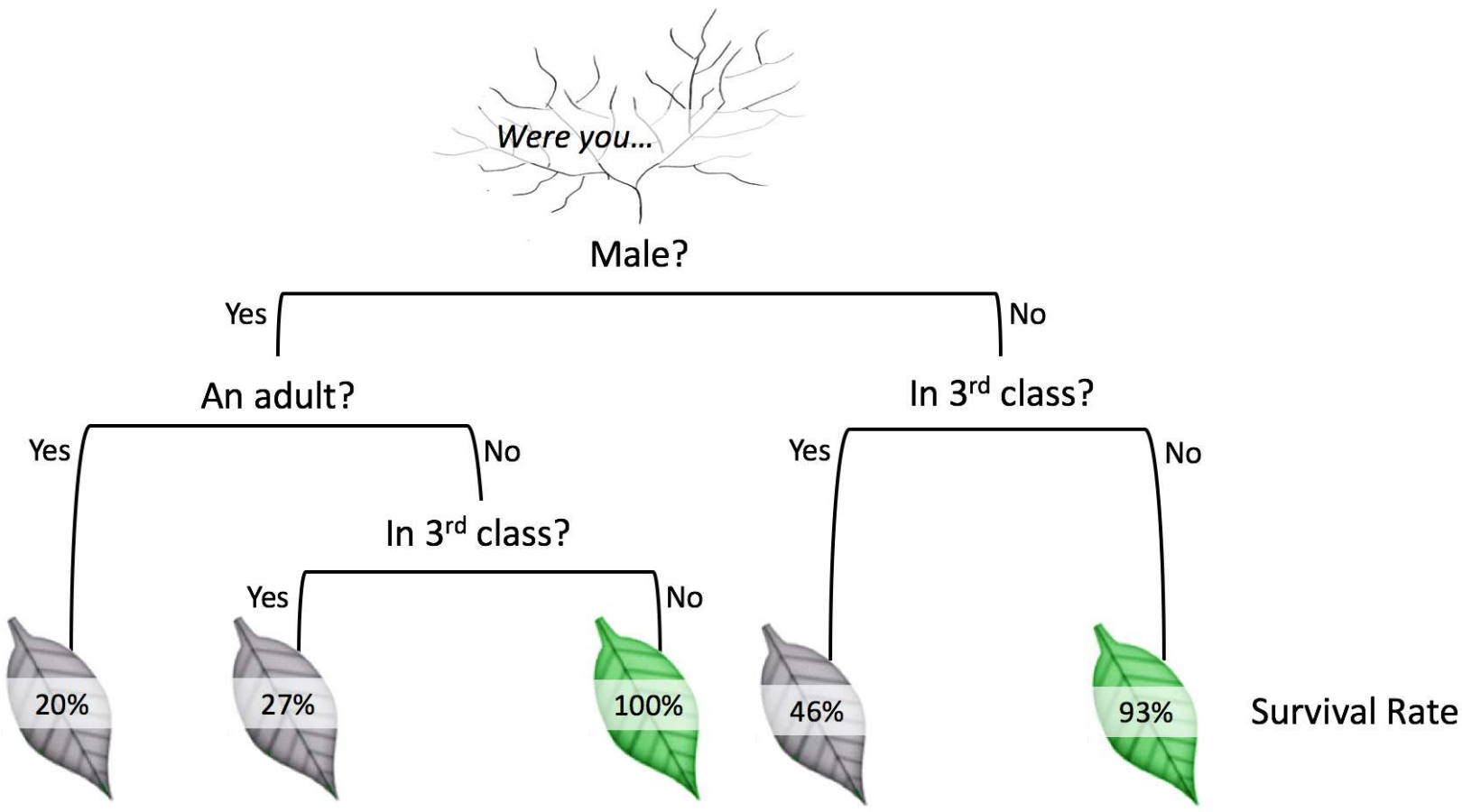
Naive Bayes is a simple, yet effective and commonly-used, machine learning classifier. It is a probabilistic classifier that makes classifications using the Maximum A Posteriori decision rule in a Bayesian setting. It can also be represented using a very simple Bayesian network. Naive Bayes classifiers have been especially popular for text classification, and are a traditional solution for problems such as spam detection.

In [29]:
```
nb_model = GaussianNB()
nb_model.fit(x_train,y_train)
nb_predict = nb_model.predict(x_test)

print('Naive bayes - ',accuracy_score(nb_predict,y_test))
```

```
Naive bayes -  0.9555555555555556
```

Decision tree



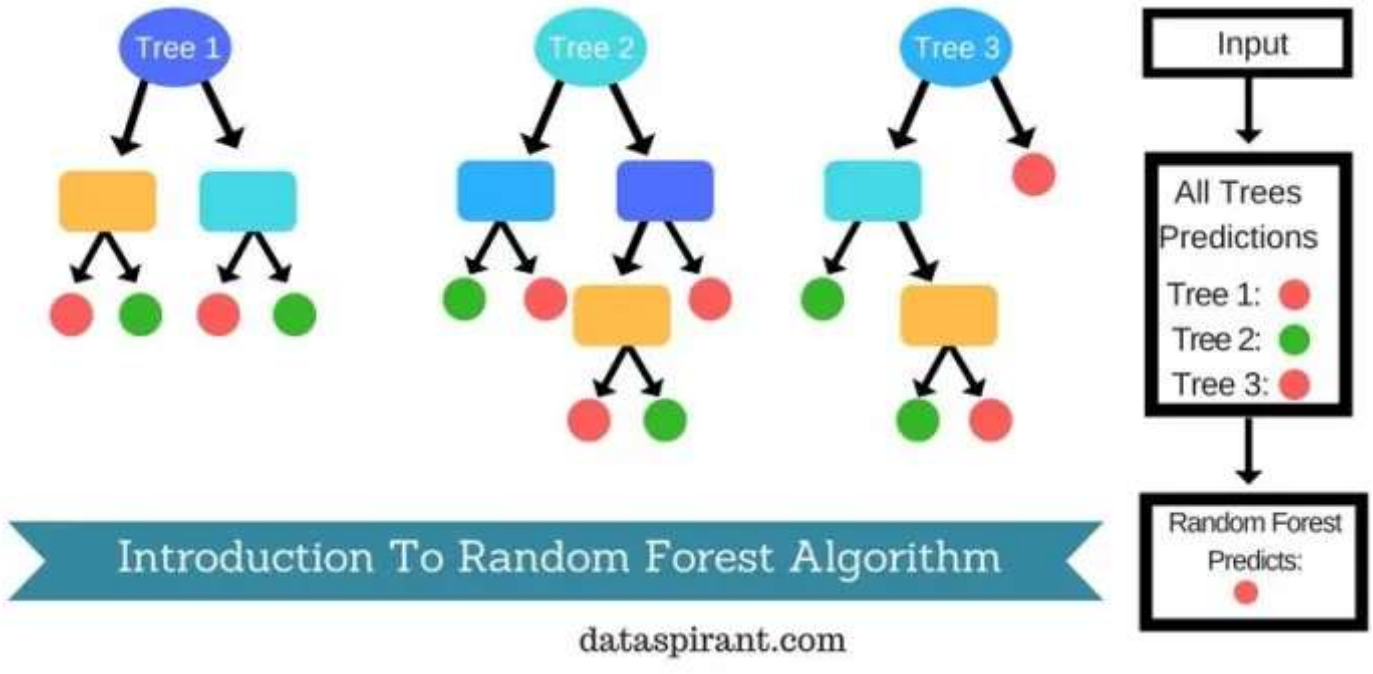
Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.

In [30]:
```
dt_model = DecisionTreeClassifier(max_leaf_nodes=3)
dt_model.fit(x_train,y_train)
dt_predict = dt_model.predict(x_test)

print('Decision Tree - ',accuracy_score(dt_predict,y_test))
```

```
Decision Tree -  0.9333333333333333
```

Random forest



Random Forest is considered to be a panacea of all data science problems. On a funny note, when you can't think of any algorithm (irrespective of situation), use random forest!

Random Forest is a versatile machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration, and does a fairly good job. It is a type of ensemble learning method, where a group of weak models combine to form a powerful model.

In [31]:
```
rfc_model = RandomForestClassifier(max_depth=3)
rfc_model.fit(x_train,y_train)
rfc_predict = rfc_model.predict(x_test)

print('Random Forest - ',accuracy_score(rfc_predict,y_test))
```

```
Random Forest -  0.9555555555555556
```

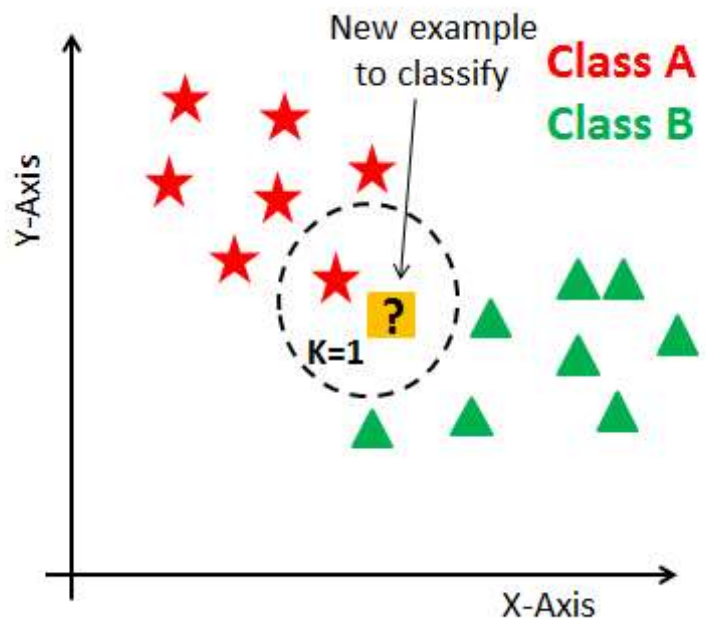
Extra Tree Classifier

In [32]:
```
etc_model = ExtraTreesClassifier()
etc_model.fit(x_train,y_train)
etc_predict = etc_model.predict(x_test)

print('Extra Tree Classifier - ',accuracy_score(etc_predict,y_test))
```

```
Extra Tree Classifier -  0.9777777777777777
```

KNN



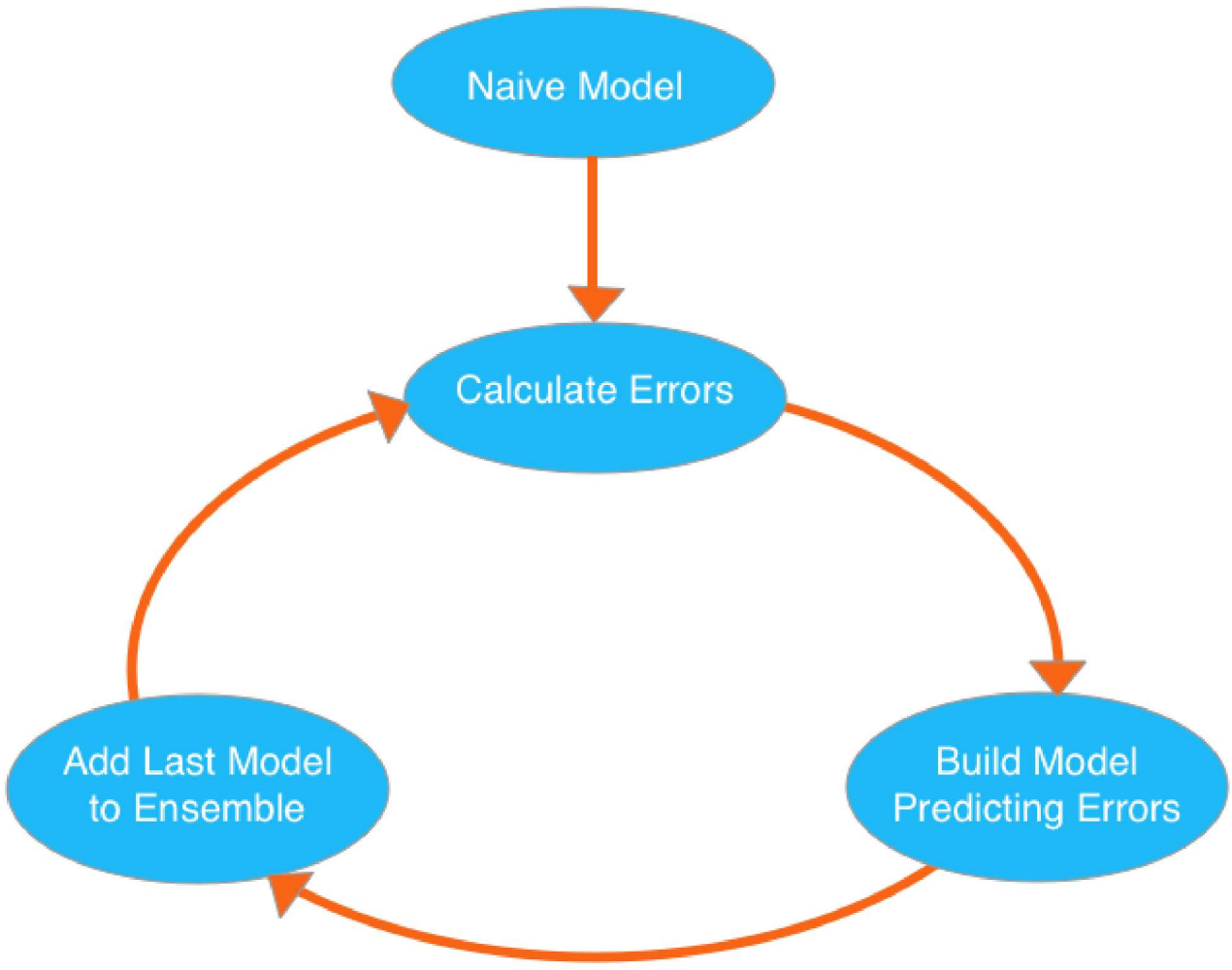
K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

In [33]:
```
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(x_train,y_train)
knn_predict = knn_model.predict(x_test)

print('knn - ',accuracy_score(knn_predict,y_test))
```

```
knn -  1.0
```

XGBoost



The beauty of this powerful algorithm lies in its scalability, which drives fast learning through parallel and distributed computing and offers efficient memory usage.

It's no wonder then that CERN recognized it as the best approach to classify signals from the Large Hadron Collider. This particular challenge posed by CERN required a solution that would be scalable to process data being generated at the rate of 3 petabytes per year and effectively distinguish an extremely rare signal from background noises in a complex physical process. XGBoost emerged as the most useful, straightforward and robust solution.

In [34]:
```
xg_model = xgb.XGBClassifier()
xg_model = xg_model.fit(x_train,y_train)
xg_model.score(x_test, y_test)
```

Out[34]:
```
0.9777777777777777
```

Deep Learning

In [35]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [36]:
```
from sklearn.preprocessing import StandardScaler, LabelBinarizer
X = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = df['Species']

X = StandardScaler().fit_transform(X)
y = LabelBinarizer().fit_transform(y)
```

Spliting the data into train - 70% and test - 30%

In [32]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 101)
```

Shallow Deep learning

In [33]:
```
shallow_model = Sequential()
shallow_model.add(Dense( 4, input_dim=4, activation = 'relu'))
shallow_model.add(Dense( units = 10, activation= 'relu'))
shallow_model.add(Dense( units = 3, activation= 'softmax'))
shallow_model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

In [34]:
```
shallow_history = shallow_model.fit(x_train, y_train, epochs = 150, validation_data = (x_test, y_test))
```

```
Train on 105 samples, validate on 45 samples
Epoch 1/150
105/105 [==============================] - 3s 32ms/step - loss: 0.5866 - acc: 0.7175 - val_loss: 0.6545 - val_acc: 0.68
15
Epoch 2/150
105/105 [==============================] - 0s 134us/step - loss: 0.5818 - acc: 0.7175 - val_loss: 0.6501 - val_acc: 0.6
815
Epoch 3/150
105/105 [==============================] - 0s 126us/step - loss: 0.5767 - acc: 0.7206 - val_loss: 0.6458 - val_acc: 0.6
815
Epoch 4/150
105/105 [==============================] - 0s 133us/step - loss: 0.5717 - acc: 0.7302 - val_loss: 0.6416 - val_acc: 0.6
815
Epoch 5/150
105/105 [==============================] - 0s 134us/step - loss: 0.5666 - acc: 0.7333 - val_loss: 0.6375 - val_acc: 0.6
815
Epoch 6/150
105/105 [==============================] - 0s 138us/step - loss: 0.5611 - acc: 0.7429 - val_loss: 0.6336 - val_acc: 0.6
815
Epoch 7/150
105/105 [==============================] - 0s 136us/step - loss: 0.5561 - acc: 0.7460 - val_loss: 0.6294 - val_acc: 0.6
815
Epoch 8/150
105/105 [==============================] - 0s 140us/step - loss: 0.5510 - acc: 0.7587 - val_loss: 0.6253 - val_acc: 0.6
889
Epoch 9/150
105/105 [==============================] - 0s 142us/step - loss: 0.5461 - acc: 0.7651 - val_loss: 0.6211 - val_acc: 0.6
889
Epoch 10/150
105/105 [==============================] - 0s 123us/step - loss: 0.5409 - acc: 0.7651 - val_loss: 0.6169 - val_acc: 0.6
889
Epoch 11/150
105/105 [==============================] - 0s 137us/step - loss: 0.5359 - acc: 0.7714 - val_loss: 0.6130 - val_acc: 0.6
963
Epoch 12/150
105/105 [==============================] - 0s 135us/step - loss: 0.5309 - acc: 0.7746 - val_loss: 0.6087 - val_acc: 0.6
963
Epoch 13/150
105/105 [==============================] - 0s 135us/step - loss: 0.5258 - acc: 0.7714 - val_loss: 0.6044 - val_acc: 0.7
185
Epoch 14/150
105/105 [==============================] - 0s 142us/step - loss: 0.5204 - acc: 0.7778 - val_loss: 0.6001 - val_acc: 0.7
185
Epoch 15/150
105/105 [==============================] - 0s 125us/step - loss: 0.5150 - acc: 0.7810 - val_loss: 0.5957 - val_acc: 0.7
259
Epoch 16/150
105/105 [==============================] - 0s 124us/step - loss: 0.5098 - acc: 0.7810 - val_loss: 0.5916 - val_acc: 0.7
333
Epoch 17/150
105/105 [==============================] - 0s 130us/step - loss: 0.5043 - acc: 0.7841 - val_loss: 0.5874 - val_acc: 0.7
333
Epoch 18/150
105/105 [==============================] - 0s 128us/step - loss: 0.4987 - acc: 0.7873 - val_loss: 0.5829 - val_acc: 0.7
333
Epoch 19/150
105/105 [==============================] - 0s 121us/step - loss: 0.4934 - acc: 0.7841 - val_loss: 0.5784 - val_acc: 0.7
333
Epoch 20/150
105/105 [==============================] - 0s 129us/step - loss: 0.4881 - acc: 0.7937 - val_loss: 0.5738 - val_acc: 0.7
259
Epoch 21/150
105/105 [==============================] - 0s 133us/step - loss: 0.4824 - acc: 0.7968 - val_loss: 0.5689 - val_acc: 0.7
185
Epoch 22/150
105/105 [==============================] - 0s 124us/step - loss: 0.4771 - acc: 0.7968 - val_loss: 0.5636 - val_acc: 0.7
333
Epoch 23/150
105/105 [==============================] - 0s 125us/step - loss: 0.4719 - acc: 0.7968 - val_loss: 0.5585 - val_acc: 0.7
333
Epoch 24/150
105/105 [==============================] - 0s 144us/step - loss: 0.4665 - acc: 0.8000 - val_loss: 0.5533 - val_acc: 0.7
407
Epoch 25/150
105/105 [==============================] - 0s 136us/step - loss: 0.4614 - acc: 0.8000 - val_loss: 0.5481 - val_acc: 0.7
481
Epoch 26/150
105/105 [==============================] - 0s 135us/step - loss: 0.4562 - acc: 0.8000 - val_loss: 0.5429 - val_acc: 0.7
481
Epoch 27/150
105/105 [==============================] - 0s 136us/step - loss: 0.4509 - acc: 0.8000 - val_loss: 0.5380 - val_acc: 0.7
407
Epoch 28/150
105/105 [==============================] - 0s 128us/step - loss: 0.4457 - acc: 0.8032 - val_loss: 0.5333 - val_acc: 0.7
407
Epoch 29/150
105/105 [==============================] - 0s 138us/step - loss: 0.4407 - acc: 0.8095 - val_loss: 0.5287 - val_acc: 0.7
407
Epoch 30/150
105/105 [==============================] - 0s 131us/step - loss: 0.4356 - acc: 0.8095 - val_loss: 0.5241 - val_acc: 0.7
481
Epoch 31/150
105/105 [==============================] - 0s 126us/step - loss: 0.4308 - acc: 0.8095 - val_loss: 0.5195 - val_acc: 0.7
481
Epoch 32/150
```

```
105/105 [==============================] - 0s 121us/step - loss: 0.4260 - acc: 0.8095 - val_loss: 0.5148 - val_acc: 0.7
481
Epoch 33/150
105/105 [==============================] - 0s 128us/step - loss: 0.4215 - acc: 0.8095 - val_loss: 0.5100 - val_acc: 0.7
481
Epoch 34/150
105/105 [==============================] - 0s 120us/step - loss: 0.4170 - acc: 0.8159 - val_loss: 0.5053 - val_acc: 0.7
481
Epoch 35/150
105/105 [==============================] - 0s 137us/step - loss: 0.4131 - acc: 0.8159 - val_loss: 0.5004 - val_acc: 0.7
481
Epoch 36/150
105/105 [==============================] - 0s 140us/step - loss: 0.4086 - acc: 0.8159 - val_loss: 0.4957 - val_acc: 0.7
481
Epoch 37/150
105/105 [==============================] - 0s 137us/step - loss: 0.4044 - acc: 0.8190 - val_loss: 0.4915 - val_acc: 0.7
481
Epoch 38/150
105/105 [==============================] - 0s 125us/step - loss: 0.4004 - acc: 0.8190 - val_loss: 0.4872 - val_acc: 0.7
556
Epoch 39/150
105/105 [==============================] - 0s 125us/step - loss: 0.3963 - acc: 0.8190 - val_loss: 0.4830 - val_acc: 0.7
556
Epoch 40/150
105/105 [==============================] - 0s 122us/step - loss: 0.3924 - acc: 0.8222 - val_loss: 0.4789 - val_acc: 0.7
630
Epoch 41/150
105/105 [==============================] - 0s 121us/step - loss: 0.3888 - acc: 0.8222 - val_loss: 0.4749 - val_acc: 0.7
630
Epoch 42/150
105/105 [==============================] - 0s 132us/step - loss: 0.3852 - acc: 0.8222 - val_loss: 0.4707 - val_acc: 0.7
630
Epoch 43/150
105/105 [==============================] - 0s 120us/step - loss: 0.3819 - acc: 0.8317 - val_loss: 0.4664 - val_acc: 0.7
630
Epoch 44/150
105/105 [==============================] - 0s 134us/step - loss: 0.3786 - acc: 0.8317 - val_loss: 0.4625 - val_acc: 0.7
630
Epoch 45/150
105/105 [==============================] - 0s 132us/step - loss: 0.3753 - acc: 0.8349 - val_loss: 0.4585 - val_acc: 0.7
630
Epoch 46/150
105/105 [==============================] - 0s 138us/step - loss: 0.3723 - acc: 0.8381 - val_loss: 0.4546 - val_acc: 0.7
704
Epoch 47/150
105/105 [==============================] - 0s 136us/step - loss: 0.3692 - acc: 0.8413 - val_loss: 0.4507 - val_acc: 0.7
704
Epoch 48/150
105/105 [==============================] - 0s 126us/step - loss: 0.3662 - acc: 0.8413 - val_loss: 0.4469 - val_acc: 0.7
704
Epoch 49/150
105/105 [==============================] - 0s 136us/step - loss: 0.3633 - acc: 0.8413 - val_loss: 0.4434 - val_acc: 0.7
704
Epoch 50/150
105/105 [==============================] - 0s 121us/step - loss: 0.3604 - acc: 0.8413 - val_loss: 0.4402 - val_acc: 0.7
704
Epoch 51/150
105/105 [==============================] - 0s 121us/step - loss: 0.3577 - acc: 0.8381 - val_loss: 0.4369 - val_acc: 0.7
704
Epoch 52/150
105/105 [==============================] - 0s 138us/step - loss: 0.3549 - acc: 0.8381 - val_loss: 0.4337 - val_acc: 0.7
704
Epoch 53/150
105/105 [==============================] - 0s 126us/step - loss: 0.3525 - acc: 0.8381 - val_loss: 0.4305 - val_acc: 0.7
704
Epoch 54/150
105/105 [==============================] - 0s 135us/step - loss: 0.3499 - acc: 0.8349 - val_loss: 0.4270 - val_acc: 0.7
704
Epoch 55/150
105/105 [==============================] - 0s 128us/step - loss: 0.3474 - acc: 0.8349 - val_loss: 0.4238 - val_acc: 0.7
778
Epoch 56/150
105/105 [==============================] - 0s 128us/step - loss: 0.3449 - acc: 0.8349 - val_loss: 0.4208 - val_acc: 0.7
778
Epoch 57/150
105/105 [==============================] - 0s 121us/step - loss: 0.3424 - acc: 0.8381 - val_loss: 0.4175 - val_acc: 0.7
778
Epoch 58/150
105/105 [==============================] - 0s 128us/step - loss: 0.3401 - acc: 0.8381 - val_loss: 0.4140 - val_acc: 0.7
778
Epoch 59/150
105/105 [==============================] - 0s 126us/step - loss: 0.3376 - acc: 0.8381 - val_loss: 0.4109 - val_acc: 0.7
778
Epoch 60/150
105/105 [==============================] - 0s 124us/step - loss: 0.3353 - acc: 0.8381 - val_loss: 0.4079 - val_acc: 0.7
778
Epoch 61/150
105/105 [==============================] - 0s 139us/step - loss: 0.3329 - acc: 0.8381 - val_loss: 0.4050 - val_acc: 0.7
778
Epoch 62/150
105/105 [==============================] - 0s 126us/step - loss: 0.3307 - acc: 0.8381 - val_loss: 0.4018 - val_acc: 0.7
852
Epoch 63/150
105/105 [==============================] - 0s 125us/step - loss: 0.3283 - acc: 0.8381 - val_loss: 0.3988 - val_acc: 0.7
852
```

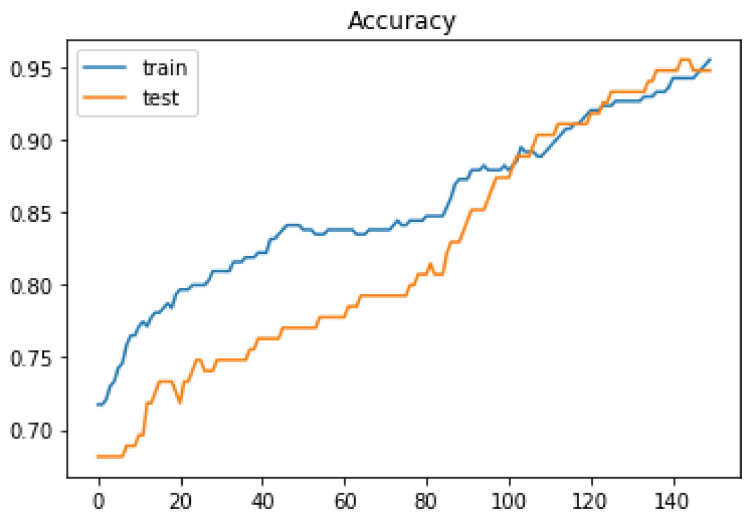
```
Epoch 64/150
105/105 [==============================] - 0s 122us/step - loss: 0.3259 - acc: 0.8349 - val_loss: 0.3959 - val_acc: 0.7
852
Epoch 65/150
105/105 [==============================] - 0s 120us/step - loss: 0.3237 - acc: 0.8349 - val_loss: 0.3931 - val_acc: 0.7
926
Epoch 66/150
105/105 [==============================] - 0s 127us/step - loss: 0.3215 - acc: 0.8349 - val_loss: 0.3901 - val_acc: 0.7
926
Epoch 67/150
105/105 [==============================] - 0s 128us/step - loss: 0.3193 - acc: 0.8381 - val_loss: 0.3873 - val_acc: 0.7
926
Epoch 68/150
105/105 [==============================] - 0s 131us/step - loss: 0.3171 - acc: 0.8381 - val_loss: 0.3844 - val_acc: 0.7
926
Epoch 69/150
105/105 [==============================] - 0s 125us/step - loss: 0.3147 - acc: 0.8381 - val_loss: 0.3815 - val_acc: 0.7
926
Epoch 70/150
105/105 [==============================] - 0s 128us/step - loss: 0.3124 - acc: 0.8381 - val_loss: 0.3785 - val_acc: 0.7
926
Epoch 71/150
105/105 [==============================] - 0s 130us/step - loss: 0.3098 - acc: 0.8381 - val_loss: 0.3753 - val_acc: 0.7
926
Epoch 72/150
105/105 [==============================] - 0s 126us/step - loss: 0.3072 - acc: 0.8381 - val_loss: 0.3722 - val_acc: 0.7
926
Epoch 73/150
105/105 [==============================] - 0s 127us/step - loss: 0.3048 - acc: 0.8413 - val_loss: 0.3691 - val_acc: 0.7
926
Epoch 74/150
105/105 [==============================] - 0s 140us/step - loss: 0.3024 - acc: 0.8444 - val_loss: 0.3662 - val_acc: 0.7
926
Epoch 75/150
105/105 [==============================] - 0s 139us/step - loss: 0.3000 - acc: 0.8413 - val_loss: 0.3633 - val_acc: 0.7
926
Epoch 76/150
105/105 [==============================] - 0s 137us/step - loss: 0.2974 - acc: 0.8413 - val_loss: 0.3605 - val_acc: 0.7
926
Epoch 77/150
105/105 [==============================] - 0s 131us/step - loss: 0.2952 - acc: 0.8444 - val_loss: 0.3576 - val_acc: 0.8
000
Epoch 78/150
105/105 [==============================] - 0s 123us/step - loss: 0.2929 - acc: 0.8444 - val_loss: 0.3548 - val_acc: 0.8
000
Epoch 79/150
105/105 [==============================] - 0s 127us/step - loss: 0.2905 - acc: 0.8444 - val_loss: 0.3522 - val_acc: 0.8
074
Epoch 80/150
105/105 [==============================] - 0s 122us/step - loss: 0.2884 - acc: 0.8444 - val_loss: 0.3497 - val_acc: 0.8
074
Epoch 81/150
105/105 [==============================] - 0s 127us/step - loss: 0.2864 - acc: 0.8476 - val_loss: 0.3475 - val_acc: 0.8
074
Epoch 82/150
105/105 [==============================] - 0s 133us/step - loss: 0.2846 - acc: 0.8476 - val_loss: 0.3450 - val_acc: 0.8
148
Epoch 83/150
105/105 [==============================] - 0s 128us/step - loss: 0.2824 - acc: 0.8476 - val_loss: 0.3420 - val_acc: 0.8
074
Epoch 84/150
105/105 [==============================] - 0s 139us/step - loss: 0.2802 - acc: 0.8476 - val_loss: 0.3391 - val_acc: 0.8
074
Epoch 85/150
105/105 [==============================] - 0s 138us/step - loss: 0.2781 - acc: 0.8476 - val_loss: 0.3363 - val_acc: 0.8
074
Epoch 86/150
105/105 [==============================] - 0s 141us/step - loss: 0.2762 - acc: 0.8540 - val_loss: 0.3334 - val_acc: 0.8
222
Epoch 87/150
105/105 [==============================] - 0s 137us/step - loss: 0.2742 - acc: 0.8603 - val_loss: 0.3307 - val_acc: 0.8
296
Epoch 88/150
105/105 [==============================] - 0s 125us/step - loss: 0.2723 - acc: 0.8698 - val_loss: 0.3280 - val_acc: 0.8
296
Epoch 89/150
105/105 [==============================] - 0s 149us/step - loss: 0.2703 - acc: 0.8730 - val_loss: 0.3255 - val_acc: 0.8
296
Epoch 90/150
105/105 [==============================] - 0s 132us/step - loss: 0.2684 - acc: 0.8730 - val_loss: 0.3232 - val_acc: 0.8
370
Epoch 91/150
105/105 [==============================] - 0s 130us/step - loss: 0.2665 - acc: 0.8730 - val_loss: 0.3205 - val_acc: 0.8
444
Epoch 92/150
105/105 [==============================] - 0s 135us/step - loss: 0.2645 - acc: 0.8794 - val_loss: 0.3182 - val_acc: 0.8
519
Epoch 93/150
105/105 [==============================] - 0s 122us/step - loss: 0.2628 - acc: 0.8794 - val_loss: 0.3159 - val_acc: 0.8
519
Epoch 94/150
105/105 [==============================] - 0s 128us/step - loss: 0.2608 - acc: 0.8794 - val_loss: 0.3136 - val_acc: 0.8
519
Epoch 95/150
105/105 [==============================] - 0s 142us/step - loss: 0.2590 - acc: 0.8825 - val_loss: 0.3115 - val_acc: 0.8
```

```
519
Epoch 96/150
105/105 [==============================] - 0s 129us/step - loss: 0.2571 - acc: 0.8794 - val_loss: 0.3091 - val_acc: 0.8
593
Epoch 97/150
105/105 [==============================] - 0s 139us/step - loss: 0.2553 - acc: 0.8794 - val_loss: 0.3069 - val_acc: 0.8
667
Epoch 98/150
105/105 [==============================] - 0s 127us/step - loss: 0.2534 - acc: 0.8794 - val_loss: 0.3046 - val_acc: 0.8
741
Epoch 99/150
105/105 [==============================] - 0s 133us/step - loss: 0.2516 - acc: 0.8794 - val_loss: 0.3023 - val_acc: 0.8
741
Epoch 100/150
105/105 [==============================] - 0s 129us/step - loss: 0.2497 - acc: 0.8825 - val_loss: 0.2999 - val_acc: 0.8
741
Epoch 101/150
105/105 [==============================] - 0s 133us/step - loss: 0.2478 - acc: 0.8794 - val_loss: 0.2975 - val_acc: 0.8
741
Epoch 102/150
105/105 [==============================] - 0s 140us/step - loss: 0.2460 - acc: 0.8825 - val_loss: 0.2952 - val_acc: 0.8
815
Epoch 103/150
105/105 [==============================] - 0s 128us/step - loss: 0.2443 - acc: 0.8857 - val_loss: 0.2928 - val_acc: 0.8
889
Epoch 104/150
105/105 [==============================] - 0s 125us/step - loss: 0.2423 - acc: 0.8952 - val_loss: 0.2906 - val_acc: 0.8
889
Epoch 105/150
105/105 [==============================] - 0s 124us/step - loss: 0.2407 - acc: 0.8921 - val_loss: 0.2884 - val_acc: 0.8
889
Epoch 106/150
105/105 [==============================] - 0s 123us/step - loss: 0.2388 - acc: 0.8921 - val_loss: 0.2864 - val_acc: 0.8
889
Epoch 107/150
105/105 [==============================] - 0s 131us/step - loss: 0.2370 - acc: 0.8921 - val_loss: 0.2844 - val_acc: 0.8
963
Epoch 108/150
105/105 [==============================] - 0s 128us/step - loss: 0.2352 - acc: 0.8889 - val_loss: 0.2825 - val_acc: 0.9
037
Epoch 109/150
105/105 [==============================] - 0s 124us/step - loss: 0.2338 - acc: 0.8889 - val_loss: 0.2803 - val_acc: 0.9
037
Epoch 110/150
105/105 [==============================] - 0s 125us/step - loss: 0.2317 - acc: 0.8921 - val_loss: 0.2778 - val_acc: 0.9
037
Epoch 111/150
105/105 [==============================] - 0s 128us/step - loss: 0.2300 - acc: 0.8952 - val_loss: 0.2752 - val_acc: 0.9
037
Epoch 112/150
105/105 [==============================] - 0s 146us/step - loss: 0.2281 - acc: 0.8984 - val_loss: 0.2728 - val_acc: 0.9
037
Epoch 113/150
105/105 [==============================] - 0s 136us/step - loss: 0.2264 - acc: 0.9016 - val_loss: 0.2706 - val_acc: 0.9
111
Epoch 114/150
105/105 [==============================] - 0s 128us/step - loss: 0.2248 - acc: 0.9048 - val_loss: 0.2682 - val_acc: 0.9
111
Epoch 115/150
105/105 [==============================] - 0s 134us/step - loss: 0.2232 - acc: 0.9079 - val_loss: 0.2660 - val_acc: 0.9
111
Epoch 116/150
105/105 [==============================] - 0s 121us/step - loss: 0.2216 - acc: 0.9079 - val_loss: 0.2637 - val_acc: 0.9
111
Epoch 117/150
105/105 [==============================] - 0s 140us/step - loss: 0.2200 - acc: 0.9111 - val_loss: 0.2616 - val_acc: 0.9
111
Epoch 118/150
105/105 [==============================] - 0s 149us/step - loss: 0.2186 - acc: 0.9111 - val_loss: 0.2593 - val_acc: 0.9
111
Epoch 119/150
105/105 [==============================] - 0s 138us/step - loss: 0.2169 - acc: 0.9143 - val_loss: 0.2573 - val_acc: 0.9
111
Epoch 120/150
105/105 [==============================] - 0s 148us/step - loss: 0.2154 - acc: 0.9175 - val_loss: 0.2552 - val_acc: 0.9
111
Epoch 121/150
105/105 [==============================] - 0s 139us/step - loss: 0.2136 - acc: 0.9206 - val_loss: 0.2532 - val_acc: 0.9
185
Epoch 122/150
105/105 [==============================] - 0s 128us/step - loss: 0.2122 - acc: 0.9206 - val_loss: 0.2510 - val_acc: 0.9
185
Epoch 123/150
105/105 [==============================] - 0s 134us/step - loss: 0.2105 - acc: 0.9206 - val_loss: 0.2490 - val_acc: 0.9
185
Epoch 124/150
105/105 [==============================] - 0s 139us/step - loss: 0.2089 - acc: 0.9238 - val_loss: 0.2468 - val_acc: 0.9
259
Epoch 125/150
105/105 [==============================] - 0s 131us/step - loss: 0.2070 - acc: 0.9238 - val_loss: 0.2441 - val_acc: 0.9
259
Epoch 126/150
105/105 [==============================] - 0s 123us/step - loss: 0.2050 - acc: 0.9238 - val_loss: 0.2413 - val_acc: 0.9
333
Epoch 127/150
```

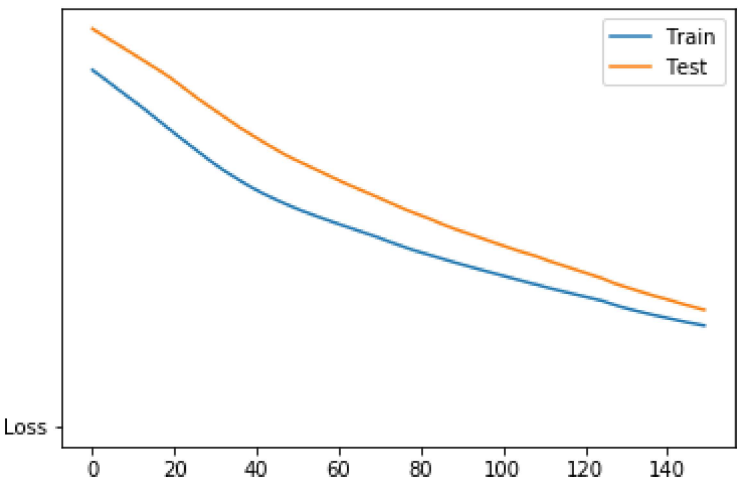
```
105/105 [==============================] - 0s 128us/step - loss: 0.2024 - acc: 0.9270 - val_loss: 0.2385 - val_acc: 0.9
333
Epoch 128/150
105/105 [==============================] - 0s 129us/step - loss: 0.2006 - acc: 0.9270 - val_loss: 0.2357 - val_acc: 0.9
333
Epoch 129/150
105/105 [==============================] - 0s 124us/step - loss: 0.1981 - acc: 0.9270 - val_loss: 0.2335 - val_acc: 0.9
333
Epoch 130/150
105/105 [==============================] - 0s 127us/step - loss: 0.1962 - acc: 0.9270 - val_loss: 0.2312 - val_acc: 0.9
333
Epoch 131/150
105/105 [==============================] - 0s 134us/step - loss: 0.1944 - acc: 0.9270 - val_loss: 0.2291 - val_acc: 0.9
333
Epoch 132/150
105/105 [==============================] - 0s 127us/step - loss: 0.1926 - acc: 0.9270 - val_loss: 0.2269 - val_acc: 0.9
333
Epoch 133/150
105/105 [==============================] - 0s 118us/step - loss: 0.1909 - acc: 0.9270 - val_loss: 0.2249 - val_acc: 0.9
333
Epoch 134/150
105/105 [==============================] - 0s 123us/step - loss: 0.1892 - acc: 0.9302 - val_loss: 0.2228 - val_acc: 0.9
333
Epoch 135/150
105/105 [==============================] - 0s 125us/step - loss: 0.1877 - acc: 0.9302 - val_loss: 0.2205 - val_acc: 0.9
407
Epoch 136/150
105/105 [==============================] - 0s 123us/step - loss: 0.1860 - acc: 0.9302 - val_loss: 0.2184 - val_acc: 0.9
407
Epoch 137/150
105/105 [==============================] - 0s 131us/step - loss: 0.1843 - acc: 0.9333 - val_loss: 0.2165 - val_acc: 0.9
481
Epoch 138/150
105/105 [==============================] - 0s 137us/step - loss: 0.1828 - acc: 0.9333 - val_loss: 0.2145 - val_acc: 0.9
481
Epoch 139/150
105/105 [==============================] - 0s 137us/step - loss: 0.1811 - acc: 0.9333 - val_loss: 0.2128 - val_acc: 0.9
481
Epoch 140/150
105/105 [==============================] - 0s 141us/step - loss: 0.1797 - acc: 0.9365 - val_loss: 0.2110 - val_acc: 0.9
481
Epoch 141/150
105/105 [==============================] - 0s 150us/step - loss: 0.1783 - acc: 0.9429 - val_loss: 0.2091 - val_acc: 0.9
481
Epoch 142/150
105/105 [==============================] - 0s 127us/step - loss: 0.1767 - acc: 0.9429 - val_loss: 0.2070 - val_acc: 0.9
481
Epoch 143/150
105/105 [==============================] - 0s 145us/step - loss: 0.1752 - acc: 0.9429 - val_loss: 0.2049 - val_acc: 0.9
556
Epoch 144/150
105/105 [==============================] - 0s 132us/step - loss: 0.1738 - acc: 0.9429 - val_loss: 0.2029 - val_acc: 0.9
556
Epoch 145/150
105/105 [==============================] - 0s 123us/step - loss: 0.1725 - acc: 0.9429 - val_loss: 0.2010 - val_acc: 0.9
556
Epoch 146/150
105/105 [==============================] - 0s 118us/step - loss: 0.1711 - acc: 0.9429 - val_loss: 0.1991 - val_acc: 0.9
481
Epoch 147/150
105/105 [==============================] - 0s 124us/step - loss: 0.1700 - acc: 0.9460 - val_loss: 0.1972 - val_acc: 0.9
481
Epoch 148/150
105/105 [==============================] - 0s 187us/step - loss: 0.1685 - acc: 0.9492 - val_loss: 0.1954 - val_acc: 0.9
481
Epoch 149/150
105/105 [==============================] - 0s 193us/step - loss: 0.1673 - acc: 0.9524 - val_loss: 0.1936 - val_acc: 0.9
481
Epoch 150/150
105/105 [==============================] - 0s 179us/step - loss: 0.1660 - acc: 0.9556 - val_loss: 0.1919 - val_acc: 0.9
481
```

In [35]:

```
plt.plot(shallow_history.history['acc'])
plt.plot(shallow_history.history['val_acc'])
plt.title("Accuracy")
plt.legend(['train', 'test'])
plt.show()
```



In [36]:
```
plt.plot(shallow_history.history['loss'])
plt.plot(shallow_history.history['val_loss'])
plt.plot('Loss')
plt.legend(['Train','Test'])
plt.show()
```



So our shallow model is good accurate.

Deep Deep learning

In [37]:
```
deep_model = Sequential()
deep_model.add(Dense( 4, input_dim=4, activation = 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 10, activation= 'relu'))
deep_model.add(Dense( units = 3, activation= 'softmax'))
deep_model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

In [38]:
```
deep_history = deep_model.fit(x_train, y_train, epochs = 150, validation_data = (x_test, y_test))
```

```
Train on 105 samples, validate on 45 samples
Epoch 1/150
105/105 [==============================] - 1s 6ms/step - loss: 1.1071 - acc: 0.1333 - val_loss: 1.1030 - val_acc: 0.044
4
Epoch 2/150
105/105 [==============================] - 0s 216us/step - loss: 1.1018 - acc: 0.1429 - val_loss: 1.1000 - val_acc: 0.2
667
Epoch 3/150
105/105 [==============================] - 0s 184us/step - loss: 1.0987 - acc: 0.3619 - val_loss: 1.0991 - val_acc: 0.2
667
Epoch 4/150
105/105 [==============================] - 0s 191us/step - loss: 1.0970 - acc: 0.3619 - val_loss: 1.0989 - val_acc: 0.2
667
Epoch 5/150
105/105 [==============================] - 0s 179us/step - loss: 1.0955 - acc: 0.3619 - val_loss: 1.0983 - val_acc: 0.3
556
Epoch 6/150
105/105 [==============================] - 0s 178us/step - loss: 1.0938 - acc: 0.5143 - val_loss: 1.0971 - val_acc: 0.5
111
Epoch 7/150
105/105 [==============================] - 0s 182us/step - loss: 1.0920 - acc: 0.6857 - val_loss: 1.0953 - val_acc: 0.5
556
Epoch 8/150
105/105 [==============================] - 0s 182us/step - loss: 1.0894 - acc: 0.7048 - val_loss: 1.0937 - val_acc: 0.5
556
Epoch 9/150
105/105 [==============================] - 0s 190us/step - loss: 1.0865 - acc: 0.7048 - val_loss: 1.0908 - val_acc: 0.5
556
Epoch 10/150
105/105 [==============================] - 0s 177us/step - loss: 1.0827 - acc: 0.7048 - val_loss: 1.0874 - val_acc: 0.5
556
Epoch 11/150
105/105 [==============================] - 0s 183us/step - loss: 1.0783 - acc: 0.7143 - val_loss: 1.0833 - val_acc: 0.5
556
Epoch 12/150
105/105 [==============================] - 0s 177us/step - loss: 1.0728 - acc: 0.7143 - val_loss: 1.0787 - val_acc: 0.5
556
Epoch 13/150
105/105 [==============================] - 0s 177us/step - loss: 1.0656 - acc: 0.7143 - val_loss: 1.0726 - val_acc: 0.5
556
Epoch 14/150
105/105 [==============================] - 0s 187us/step - loss: 1.0568 - acc: 0.7143 - val_loss: 1.0638 - val_acc: 0.5
556
Epoch 15/150
105/105 [==============================] - 0s 191us/step - loss: 1.0450 - acc: 0.7048 - val_loss: 1.0525 - val_acc: 0.5
556
Epoch 16/150
105/105 [==============================] - 0s 185us/step - loss: 1.0298 - acc: 0.7048 - val_loss: 1.0379 - val_acc: 0.5
556
Epoch 17/150
105/105 [==============================] - 0s 177us/step - loss: 1.0107 - acc: 0.7048 - val_loss: 1.0207 - val_acc: 0.5
556
Epoch 18/150
105/105 [==============================] - 0s 188us/step - loss: 0.9880 - acc: 0.7048 - val_loss: 1.0001 - val_acc: 0.5
556
Epoch 19/150
105/105 [==============================] - 0s 181us/step - loss: 0.9560 - acc: 0.7048 - val_loss: 0.9748 - val_acc: 0.5
556
Epoch 20/150
105/105 [==============================] - 0s 205us/step - loss: 0.9233 - acc: 0.7048 - val_loss: 0.9443 - val_acc: 0.5
556
Epoch 21/150
105/105 [==============================] - 0s 189us/step - loss: 0.8793 - acc: 0.7143 - val_loss: 0.9078 - val_acc: 0.5
778
Epoch 22/150
105/105 [==============================] - 0s 174us/step - loss: 0.8347 - acc: 0.7524 - val_loss: 0.8675 - val_acc: 0.6
889
Epoch 23/150
105/105 [==============================] - 0s 179us/step - loss: 0.7869 - acc: 0.7619 - val_loss: 0.8293 - val_acc: 0.6
889
Epoch 24/150
105/105 [==============================] - 0s 195us/step - loss: 0.7424 - acc: 0.7714 - val_loss: 0.7963 - val_acc: 0.7
111
Epoch 25/150
105/105 [==============================] - 0s 181us/step - loss: 0.7002 - acc: 0.7619 - val_loss: 0.7671 - val_acc: 0.7
111
Epoch 26/150
105/105 [==============================] - 0s 191us/step - loss: 0.6643 - acc: 0.7619 - val_loss: 0.7382 - val_acc: 0.7
111
Epoch 27/150
105/105 [==============================] - 0s 186us/step - loss: 0.6262 - acc: 0.7429 - val_loss: 0.7086 - val_acc: 0.7
111
Epoch 28/150
105/105 [==============================] - 0s 184us/step - loss: 0.5857 - acc: 0.7524 - val_loss: 0.6790 - val_acc: 0.6
889
Epoch 29/150
105/105 [==============================] - 0s 185us/step - loss: 0.5486 - acc: 0.7714 - val_loss: 0.6540 - val_acc: 0.7
333
Epoch 30/150
105/105 [==============================] - 0s 185us/step - loss: 0.5114 - acc: 0.7714 - val_loss: 0.6355 - val_acc: 0.7
333
Epoch 31/150
105/105 [==============================] - 0s 178us/step - loss: 0.4834 - acc: 0.8000 - val_loss: 0.6171 - val_acc: 0.7
778
Epoch 32/150
```

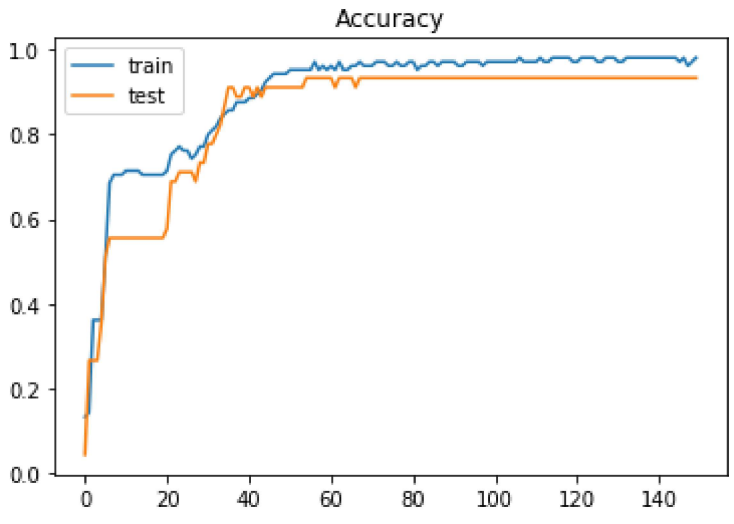
```
105/105 [==============================] - 0s 180us/step - loss: 0.4596 - acc: 0.8095 - val_loss: 0.5984 - val_acc: 0.7
778
Epoch 33/150
105/105 [==============================] - 0s 185us/step - loss: 0.4396 - acc: 0.8190 - val_loss: 0.5819 - val_acc: 0.8
000
Epoch 34/150
105/105 [==============================] - 0s 176us/step - loss: 0.4216 - acc: 0.8381 - val_loss: 0.5675 - val_acc: 0.8
222
Epoch 35/150
105/105 [==============================] - 0s 191us/step - loss: 0.4060 - acc: 0.8476 - val_loss: 0.5490 - val_acc: 0.8
667
Epoch 36/150
105/105 [==============================] - 0s 173us/step - loss: 0.3918 - acc: 0.8571 - val_loss: 0.5333 - val_acc: 0.9
111
Epoch 37/150
105/105 [==============================] - 0s 197us/step - loss: 0.3832 - acc: 0.8571 - val_loss: 0.5183 - val_acc: 0.9
111
Epoch 38/150
105/105 [==============================] - 0s 181us/step - loss: 0.3735 - acc: 0.8762 - val_loss: 0.5054 - val_acc: 0.8
889
Epoch 39/150
105/105 [==============================] - 0s 184us/step - loss: 0.3624 - acc: 0.8762 - val_loss: 0.4931 - val_acc: 0.8
889
Epoch 40/150
105/105 [==============================] - 0s 194us/step - loss: 0.3493 - acc: 0.8762 - val_loss: 0.4819 - val_acc: 0.9
111
Epoch 41/150
105/105 [==============================] - 0s 184us/step - loss: 0.3381 - acc: 0.8857 - val_loss: 0.4711 - val_acc: 0.9
111
Epoch 42/150
105/105 [==============================] - 0s 201us/step - loss: 0.3289 - acc: 0.8857 - val_loss: 0.4598 - val_acc: 0.8
889
Epoch 43/150
105/105 [==============================] - 0s 183us/step - loss: 0.3202 - acc: 0.8952 - val_loss: 0.4508 - val_acc: 0.9
111
Epoch 44/150
105/105 [==============================] - 0s 193us/step - loss: 0.3106 - acc: 0.9048 - val_loss: 0.4347 - val_acc: 0.8
889
Epoch 45/150
105/105 [==============================] - 0s 182us/step - loss: 0.3018 - acc: 0.9238 - val_loss: 0.4196 - val_acc: 0.9
111
Epoch 46/150
105/105 [==============================] - 0s 171us/step - loss: 0.2910 - acc: 0.9333 - val_loss: 0.4070 - val_acc: 0.9
111
Epoch 47/150
105/105 [==============================] - 0s 180us/step - loss: 0.2801 - acc: 0.9429 - val_loss: 0.3948 - val_acc: 0.9
111
Epoch 48/150
105/105 [==============================] - 0s 196us/step - loss: 0.2698 - acc: 0.9429 - val_loss: 0.3828 - val_acc: 0.9
111
Epoch 49/150
105/105 [==============================] - 0s 174us/step - loss: 0.2586 - acc: 0.9429 - val_loss: 0.3726 - val_acc: 0.9
111
Epoch 50/150
105/105 [==============================] - 0s 180us/step - loss: 0.2487 - acc: 0.9429 - val_loss: 0.3567 - val_acc: 0.9
111
Epoch 51/150
105/105 [==============================] - 0s 174us/step - loss: 0.2382 - acc: 0.9524 - val_loss: 0.3416 - val_acc: 0.9
111
Epoch 52/150
105/105 [==============================] - 0s 182us/step - loss: 0.2274 - acc: 0.9524 - val_loss: 0.3297 - val_acc: 0.9
111
Epoch 53/150
105/105 [==============================] - 0s 198us/step - loss: 0.2155 - acc: 0.9524 - val_loss: 0.3149 - val_acc: 0.9
111
Epoch 54/150
105/105 [==============================] - 0s 179us/step - loss: 0.2045 - acc: 0.9524 - val_loss: 0.2979 - val_acc: 0.9
111
Epoch 55/150
105/105 [==============================] - 0s 186us/step - loss: 0.1941 - acc: 0.9524 - val_loss: 0.2831 - val_acc: 0.9
333
Epoch 56/150
105/105 [==============================] - 0s 171us/step - loss: 0.1867 - acc: 0.9524 - val_loss: 0.2688 - val_acc: 0.9
333
Epoch 57/150
105/105 [==============================] - 0s 184us/step - loss: 0.1755 - acc: 0.9714 - val_loss: 0.2688 - val_acc: 0.9
333
Epoch 58/150
105/105 [==============================] - 0s 185us/step - loss: 0.1709 - acc: 0.9524 - val_loss: 0.2539 - val_acc: 0.9
333
Epoch 59/150
105/105 [==============================] - 0s 203us/step - loss: 0.1646 - acc: 0.9619 - val_loss: 0.2329 - val_acc: 0.9
333
Epoch 60/150
105/105 [==============================] - 0s 175us/step - loss: 0.1589 - acc: 0.9524 - val_loss: 0.2270 - val_acc: 0.9
333
Epoch 61/150
105/105 [==============================] - 0s 183us/step - loss: 0.1497 - acc: 0.9619 - val_loss: 0.2372 - val_acc: 0.9
333
Epoch 62/150
105/105 [==============================] - 0s 185us/step - loss: 0.1490 - acc: 0.9524 - val_loss: 0.2274 - val_acc: 0.9
111
Epoch 63/150
105/105 [==============================] - 0s 187us/step - loss: 0.1376 - acc: 0.9714 - val_loss: 0.2129 - val_acc: 0.9
333
```

```
Epoch 64/150
105/105 [==============================] - 0s 176us/step - loss: 0.1523 - acc: 0.9524 - val_loss: 0.2068 - val_acc: 0.9
333
Epoch 65/150
105/105 [==============================] - 0s 179us/step - loss: 0.1413 - acc: 0.9524 - val_loss: 0.2050 - val_acc: 0.9
333
Epoch 66/150
105/105 [==============================] - 0s 181us/step - loss: 0.1275 - acc: 0.9619 - val_loss: 0.2120 - val_acc: 0.9
333
Epoch 67/150
105/105 [==============================] - 0s 189us/step - loss: 0.1301 - acc: 0.9619 - val_loss: 0.2125 - val_acc: 0.9
111
Epoch 68/150
105/105 [==============================] - 0s 184us/step - loss: 0.1226 - acc: 0.9714 - val_loss: 0.2033 - val_acc: 0.9
333
Epoch 69/150
105/105 [==============================] - 0s 170us/step - loss: 0.1251 - acc: 0.9619 - val_loss: 0.2016 - val_acc: 0.9
333
Epoch 70/150
105/105 [==============================] - 0s 188us/step - loss: 0.1264 - acc: 0.9619 - val_loss: 0.2003 - val_acc: 0.9
333
Epoch 71/150
105/105 [==============================] - 0s 187us/step - loss: 0.1200 - acc: 0.9619 - val_loss: 0.2031 - val_acc: 0.9
333
Epoch 72/150
105/105 [==============================] - 0s 192us/step - loss: 0.1162 - acc: 0.9714 - val_loss: 0.2033 - val_acc: 0.9
333
Epoch 73/150
105/105 [==============================] - 0s 194us/step - loss: 0.1151 - acc: 0.9714 - val_loss: 0.1991 - val_acc: 0.9
333
Epoch 74/150
105/105 [==============================] - 0s 190us/step - loss: 0.1135 - acc: 0.9714 - val_loss: 0.1922 - val_acc: 0.9
333
Epoch 75/150
105/105 [==============================] - 0s 182us/step - loss: 0.1128 - acc: 0.9619 - val_loss: 0.1900 - val_acc: 0.9
333
Epoch 76/150
105/105 [==============================] - 0s 191us/step - loss: 0.1114 - acc: 0.9619 - val_loss: 0.1900 - val_acc: 0.9
333
Epoch 77/150
105/105 [==============================] - 0s 200us/step - loss: 0.1134 - acc: 0.9714 - val_loss: 0.1926 - val_acc: 0.9
333
Epoch 78/150
105/105 [==============================] - 0s 205us/step - loss: 0.1113 - acc: 0.9619 - val_loss: 0.1853 - val_acc: 0.9
333
Epoch 79/150
105/105 [==============================] - 0s 180us/step - loss: 0.1137 - acc: 0.9619 - val_loss: 0.1848 - val_acc: 0.9
333
Epoch 80/150
105/105 [==============================] - 0s 178us/step - loss: 0.1044 - acc: 0.9714 - val_loss: 0.1962 - val_acc: 0.9
333
Epoch 81/150
105/105 [==============================] - 0s 185us/step - loss: 0.1084 - acc: 0.9714 - val_loss: 0.2110 - val_acc: 0.9
333
Epoch 82/150
105/105 [==============================] - 0s 186us/step - loss: 0.1267 - acc: 0.9524 - val_loss: 0.2087 - val_acc: 0.9
333
Epoch 83/150
105/105 [==============================] - 0s 181us/step - loss: 0.1144 - acc: 0.9619 - val_loss: 0.1921 - val_acc: 0.9
333
Epoch 84/150
105/105 [==============================] - 0s 189us/step - loss: 0.1097 - acc: 0.9619 - val_loss: 0.1921 - val_acc: 0.9
333
Epoch 85/150
105/105 [==============================] - 0s 204us/step - loss: 0.1114 - acc: 0.9714 - val_loss: 0.1913 - val_acc: 0.9
333
Epoch 86/150
105/105 [==============================] - 0s 175us/step - loss: 0.1055 - acc: 0.9714 - val_loss: 0.1935 - val_acc: 0.9
333
Epoch 87/150
105/105 [==============================] - 0s 175us/step - loss: 0.1056 - acc: 0.9619 - val_loss: 0.2003 - val_acc: 0.9
333
Epoch 88/150
105/105 [==============================] - 0s 193us/step - loss: 0.1025 - acc: 0.9714 - val_loss: 0.1989 - val_acc: 0.9
333
Epoch 89/150
105/105 [==============================] - 0s 180us/step - loss: 0.1001 - acc: 0.9714 - val_loss: 0.1981 - val_acc: 0.9
333
Epoch 90/150
105/105 [==============================] - 0s 181us/step - loss: 0.0989 - acc: 0.9714 - val_loss: 0.1980 - val_acc: 0.9
333
Epoch 91/150
105/105 [==============================] - 0s 186us/step - loss: 0.0981 - acc: 0.9714 - val_loss: 0.1968 - val_acc: 0.9
333
Epoch 92/150
105/105 [==============================] - 0s 185us/step - loss: 0.0983 - acc: 0.9619 - val_loss: 0.1953 - val_acc: 0.9
333
Epoch 93/150
105/105 [==============================] - 0s 180us/step - loss: 0.0980 - acc: 0.9619 - val_loss: 0.1904 - val_acc: 0.9
333
Epoch 94/150
105/105 [==============================] - 0s 197us/step - loss: 0.1019 - acc: 0.9714 - val_loss: 0.1873 - val_acc: 0.9
333
Epoch 95/150
105/105 [==============================] - 0s 187us/step - loss: 0.1007 - acc: 0.9714 - val_loss: 0.1885 - val_acc: 0.9
```

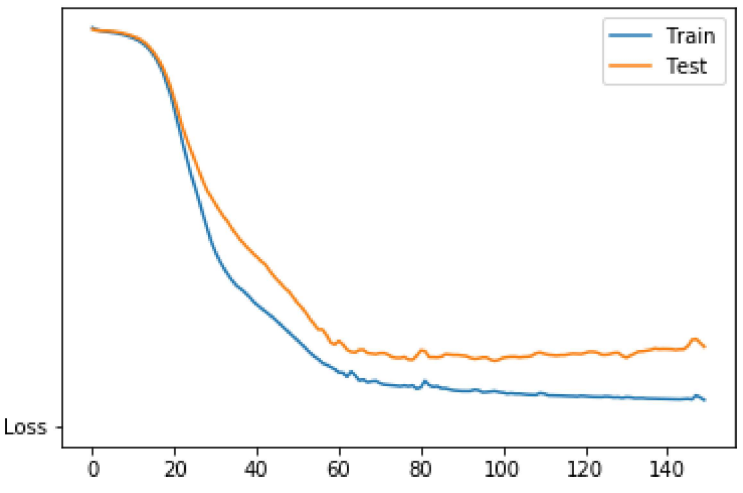
```
333
Epoch 96/150
105/105 [==============================] - 0s 187us/step - loss: 0.0946 - acc: 0.9714 - val_loss: 0.1913 - val_acc: 0.9
333
Epoch 97/150
105/105 [==============================] - 0s 189us/step - loss: 0.0952 - acc: 0.9714 - val_loss: 0.1913 - val_acc: 0.9
333
Epoch 98/150
105/105 [==============================] - 0s 187us/step - loss: 0.0968 - acc: 0.9619 - val_loss: 0.1845 - val_acc: 0.9
333
Epoch 99/150
105/105 [==============================] - 0s 186us/step - loss: 0.0981 - acc: 0.9714 - val_loss: 0.1816 - val_acc: 0.9
333
Epoch 100/150
105/105 [==============================] - 0s 189us/step - loss: 0.0951 - acc: 0.9714 - val_loss: 0.1847 - val_acc: 0.9
333
Epoch 101/150
105/105 [==============================] - 0s 196us/step - loss: 0.0936 - acc: 0.9714 - val_loss: 0.1903 - val_acc: 0.9
333
Epoch 102/150
105/105 [==============================] - 0s 177us/step - loss: 0.0906 - acc: 0.9714 - val_loss: 0.1921 - val_acc: 0.9
333
Epoch 103/150
105/105 [==============================] - 0s 187us/step - loss: 0.0920 - acc: 0.9714 - val_loss: 0.1931 - val_acc: 0.9
333
Epoch 104/150
105/105 [==============================] - 0s 197us/step - loss: 0.0904 - acc: 0.9714 - val_loss: 0.1931 - val_acc: 0.9
333
Epoch 105/150
105/105 [==============================] - 0s 188us/step - loss: 0.0906 - acc: 0.9714 - val_loss: 0.1915 - val_acc: 0.9
333
Epoch 106/150
105/105 [==============================] - 0s 185us/step - loss: 0.0892 - acc: 0.9714 - val_loss: 0.1927 - val_acc: 0.9
333
Epoch 107/150
105/105 [==============================] - 0s 191us/step - loss: 0.0881 - acc: 0.9810 - val_loss: 0.1932 - val_acc: 0.9
333
Epoch 108/150
105/105 [==============================] - 0s 180us/step - loss: 0.0880 - acc: 0.9714 - val_loss: 0.1958 - val_acc: 0.9
333
Epoch 109/150
105/105 [==============================] - 0s 190us/step - loss: 0.0873 - acc: 0.9714 - val_loss: 0.2026 - val_acc: 0.9
333
Epoch 110/150
105/105 [==============================] - 0s 183us/step - loss: 0.0925 - acc: 0.9714 - val_loss: 0.2053 - val_acc: 0.9
333
Epoch 111/150
105/105 [==============================] - 0s 184us/step - loss: 0.0905 - acc: 0.9714 - val_loss: 0.2014 - val_acc: 0.9
333
Epoch 112/150
105/105 [==============================] - 0s 192us/step - loss: 0.0861 - acc: 0.9810 - val_loss: 0.1988 - val_acc: 0.9
333
Epoch 113/150
105/105 [==============================] - 0s 180us/step - loss: 0.0859 - acc: 0.9714 - val_loss: 0.1976 - val_acc: 0.9
333
Epoch 114/150
105/105 [==============================] - 0s 171us/step - loss: 0.0857 - acc: 0.9714 - val_loss: 0.1964 - val_acc: 0.9
333
Epoch 115/150
105/105 [==============================] - 0s 192us/step - loss: 0.0851 - acc: 0.9810 - val_loss: 0.1977 - val_acc: 0.9
333
Epoch 116/150
105/105 [==============================] - 0s 169us/step - loss: 0.0844 - acc: 0.9810 - val_loss: 0.1976 - val_acc: 0.9
333
Epoch 117/150
105/105 [==============================] - 0s 178us/step - loss: 0.0838 - acc: 0.9810 - val_loss: 0.2000 - val_acc: 0.9
333
Epoch 118/150
105/105 [==============================] - 0s 196us/step - loss: 0.0837 - acc: 0.9810 - val_loss: 0.2016 - val_acc: 0.9
333
Epoch 119/150
105/105 [==============================] - 0s 179us/step - loss: 0.0829 - acc: 0.9810 - val_loss: 0.2009 - val_acc: 0.9
333
Epoch 120/150
105/105 [==============================] - 0s 178us/step - loss: 0.0845 - acc: 0.9714 - val_loss: 0.2005 - val_acc: 0.9
333
Epoch 121/150
105/105 [==============================] - 0s 188us/step - loss: 0.0837 - acc: 0.9714 - val_loss: 0.2035 - val_acc: 0.9
333
Epoch 122/150
105/105 [==============================] - 0s 175us/step - loss: 0.0833 - acc: 0.9810 - val_loss: 0.2070 - val_acc: 0.9
333
Epoch 123/150
105/105 [==============================] - 0s 188us/step - loss: 0.0819 - acc: 0.9810 - val_loss: 0.2067 - val_acc: 0.9
333
Epoch 124/150
105/105 [==============================] - 0s 188us/step - loss: 0.0819 - acc: 0.9810 - val_loss: 0.2047 - val_acc: 0.9
333
Epoch 125/150
105/105 [==============================] - 0s 197us/step - loss: 0.0816 - acc: 0.9810 - val_loss: 0.1987 - val_acc: 0.9
333
Epoch 126/150
105/105 [==============================] - 0s 198us/step - loss: 0.0829 - acc: 0.9714 - val_loss: 0.1985 - val_acc: 0.9
333
Epoch 127/150
```

```
105/105 [==============================] - 0s 203us/step - loss: 0.0822 - acc: 0.9714 - val_loss: 0.2004 - val_acc: 0.9
333
Epoch 128/150
105/105 [==============================] - 0s 199us/step - loss: 0.0804 - acc: 0.9810 - val_loss: 0.2038 - val_acc: 0.9
333
Epoch 129/150
105/105 [==============================] - 0s 189us/step - loss: 0.0811 - acc: 0.9810 - val_loss: 0.2045 - val_acc: 0.9
333
Epoch 130/150
105/105 [==============================] - 0s 196us/step - loss: 0.0789 - acc: 0.9810 - val_loss: 0.1963 - val_acc: 0.9
333
Epoch 131/150
105/105 [==============================] - 0s 181us/step - loss: 0.0812 - acc: 0.9714 - val_loss: 0.1904 - val_acc: 0.9
333
Epoch 132/150
105/105 [==============================] - 0s 188us/step - loss: 0.0806 - acc: 0.9714 - val_loss: 0.1961 - val_acc: 0.9
333
Epoch 133/150
105/105 [==============================] - 0s 183us/step - loss: 0.0783 - acc: 0.9810 - val_loss: 0.2018 - val_acc: 0.9
333
Epoch 134/150
105/105 [==============================] - 0s 177us/step - loss: 0.0783 - acc: 0.9810 - val_loss: 0.2076 - val_acc: 0.9
333
Epoch 135/150
105/105 [==============================] - 0s 192us/step - loss: 0.0786 - acc: 0.9810 - val_loss: 0.2083 - val_acc: 0.9
333
Epoch 136/150
105/105 [==============================] - 0s 188us/step - loss: 0.0774 - acc: 0.9810 - val_loss: 0.2106 - val_acc: 0.9
333
Epoch 137/150
105/105 [==============================] - 0s 209us/step - loss: 0.0772 - acc: 0.9810 - val_loss: 0.2124 - val_acc: 0.9
333
Epoch 138/150
105/105 [==============================] - 0s 185us/step - loss: 0.0774 - acc: 0.9810 - val_loss: 0.2160 - val_acc: 0.9
333
Epoch 139/150
105/105 [==============================] - 0s 191us/step - loss: 0.0767 - acc: 0.9810 - val_loss: 0.2141 - val_acc: 0.9
333
Epoch 140/150
105/105 [==============================] - 0s 193us/step - loss: 0.0766 - acc: 0.9810 - val_loss: 0.2149 - val_acc: 0.9
333
Epoch 141/150
105/105 [==============================] - 0s 200us/step - loss: 0.0762 - acc: 0.9810 - val_loss: 0.2146 - val_acc: 0.9
333
Epoch 142/150
105/105 [==============================] - 0s 238us/step - loss: 0.0756 - acc: 0.9810 - val_loss: 0.2141 - val_acc: 0.9
333
Epoch 143/150
105/105 [==============================] - 0s 185us/step - loss: 0.0755 - acc: 0.9810 - val_loss: 0.2123 - val_acc: 0.9
333
Epoch 144/150
105/105 [==============================] - 0s 183us/step - loss: 0.0751 - acc: 0.9810 - val_loss: 0.2136 - val_acc: 0.9
333
Epoch 145/150
105/105 [==============================] - 0s 184us/step - loss: 0.0762 - acc: 0.9810 - val_loss: 0.2139 - val_acc: 0.9
333
Epoch 146/150
105/105 [==============================] - 0s 171us/step - loss: 0.0767 - acc: 0.9714 - val_loss: 0.2224 - val_acc: 0.9
333
Epoch 147/150
105/105 [==============================] - 0s 179us/step - loss: 0.0748 - acc: 0.9810 - val_loss: 0.2402 - val_acc: 0.9
333
Epoch 148/150
105/105 [==============================] - 0s 249us/step - loss: 0.0857 - acc: 0.9619 - val_loss: 0.2425 - val_acc: 0.9
333
Epoch 149/150
105/105 [==============================] - 0s 263us/step - loss: 0.0806 - acc: 0.9714 - val_loss: 0.2321 - val_acc: 0.9
333
Epoch 150/150
105/105 [==============================] - 0s 245us/step - loss: 0.0732 - acc: 0.9810 - val_loss: 0.2211 - val_acc: 0.9
333
```

In [39]:
```
plt.plot(deep_history.history['acc'])
plt.plot(deep_history.history['val_acc'])
plt.title("Accuracy")
plt.legend(['train', 'test'])
plt.show()
```



In [40]:
```
plt.plot(deep_history.history['loss'])
plt.plot(deep_history.history['val_loss'])
plt.plot('Loss')
plt.legend(['Train','Test'])
plt.show()
```



So our deep model is more accurate than the shallow model.