Machine Learning with Iris dataset

Part 1. import libraries and dataset

In [ ]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn import svm

from sklearn import metrics
```

In [ ]:
```
#iris= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data/iris.csv')
iris= pd.read_csv('https://raw.githubusercontent.com/hondalee8/Intro-ML-NN/main/iris.csv')
```

Part 2. Data Analysis with Iris

In [ ]:
```
iris.shape
#150 row and 5 column
```

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

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

Statistical Summary

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

Scatter Plot of Sepal length vs width

In [ ]:
```
fig = iris[iris.Species=='Iris-setosa'].plot(kind='scatter',x='SepalLengthCm',
y='SepalWidthCm',color='orange', label='Setosa')
iris[iris.Species=='Iris-versicolor'].plot(kind='scatter',x='SepalLengthCm',y=
'SepalWidthCm',color='blue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot(kind='scatter',x='SepalLengthCm',y=
'SepalWidthCm',color='green', label='virginica', ax=fig)
fig.set_xlabel("Sepal Length")
fig.set_ylabel("Sepal Width")
fig.set_title("Sepal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
```

Scatter plot of Petal length vs width

In [ ]:
```
fig = iris[iris.Species=='Iris-setosa'].plot.scatter(x='PetalLengthCm',y='Peta
lWidthCm',color='orange', label='Setosa')
iris[iris.Species=='Iris-versicolor'].plot.scatter(x='PetalLengthCm',y='PetalW
idthCm',color='blue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot.scatter(x='PetalLengthCm',y='PetalWi
dthCm',color='green', label='virginica', ax=fig)
fig.set_xlabel("Petal Length")
fig.set_ylabel("Petal Width")
fig.set_title(" Petal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
```

Finding

As we can see that the **Petal Features** are giving a better cluster division compared to the **Sepal features**. This is an indication that the Petals can help in better and accurate Predictions over the Sepal.

In [ ]:
```
sns.pairplot(iris, hue='Species', markers='+')
plt.show()
```

Finding:

From the above, we can see that **Iris-Setosa** is separated from both other species in all the features.

The distribution of the Sepal & Petal length and width

In [ ]:
```
iris.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
plt.show()
```

Calculate the Correlation between sepal & peatal length and width

In [ ]:
```
corr= iris.corr()
corr
```

Observation :

- The Sepal Width and Length are not correlated.
- The Petal Width and Length are highly correlated

We will use all the features for training the algorithm and check the accuracy

Some ML notations

attributes-->An attribute is a property of an instance that may be used to determine its classification. In the following dataset, the attributes are the petal and sepal length and width. It is also known as Features.

Target variable, in the machine learning context is the variable that is or should be the output. Here the target variables are the 3 flower species.

Steps To Be followed When Applying an Algorithm

1. Split the dataset into training and testing dataset. The testing dataset is generally smaller than training one as it will help in training the model better
2. Select any algorithm based on the problem (classification or regression) whatever you feel may be good.
3. Then pass the training dataset to the algorithm to train it. We use the .fit() method
4. Then pass the testing data to the trained algorithm to predict the outcome. We use the .predict() method.
5. We then check the accuracy by passing the predicted outcome and the actual output to the model.

Part 3. Model selection and training

Splitting The Data into Training And Testing Dataset

In [ ]:
```
train, test = train_test_split(iris, test_size = 0.2)# in this our main data i
s split into train and test
print(train.shape)
print(test.shape)
```

In [ ]:
```
#train_X = train[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthC
m']]
train_X = train.iloc[:, 0:4]# taking the training data features: column of 'Se
palLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'
train_y=train.iloc[:, 4] # output of our training data: column of Species
test_X= test.iloc[:, 0:4] # taking test data features: column of 'SepalLengthC
m','SepalWidthCm','PetalLengthCm','PetalWidthCm'
test_y =test.iloc[:,4]   #output value of test data: column of Species
```

In [ ]:
```
train_X.head(2)
#train_y.head()
```

Which altorithm give better accuracy?

In [ ]:
```
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr'
)))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', svm.SVC(gamma='auto')))

listAcc=[]
listName =[]
for name, model in models:
    model.fit(train_X,train_y)
    prediction=model.predict(test_X)
    acc= metrics.accuracy_score(prediction,test_y)
    listAcc.append(acc)
    listName.append(name)
    print("Model={0}, Accuracy={1}".format(name, acc))
```

Choose the best algorithm for model ttraining and prediction

SVC has highest accuracy in our problem, we select this algorithm in this case. Before prediction, we first train the model by using training dataset. Then use the trained model to predict the testing dataset.

In [ ]:
```
# Make predictions on validation dataset
model = svm.SVC(gamma='auto')
model.fit(train_X, train_y) #train model
predictions = model.predict(test_X) #make predition
df_compare = pd.DataFrame({'Predict result': predictions, 'Actual ': test_y})
print(df_compare)
```

In [ ]:
```
# Evaluate predictions
print("Accuracy Score:{0}\n".format(metrics.accuracy_score(test_y, predictions
)))
print("Confusion matrix: \n {0} \n".format(metrics.confusion_matrix(test_y, pr
edictions)))
```

Appendix 1: Whole code for model building and prediction

In [ ]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics

#iris= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data/iris.csv')
iris= pd.read_csv('https://raw.githubusercontent.com/hondalee8/Intro-ML-NN/mai
n/iris.csv')

#split data to 2 parts
train, test = train_test_split(iris, test_size = 0.2)

#first 4 column is input features, last column is the label(species)
#train_X = train[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthC
m']]
train_X = train.iloc[:, 0:4]
train_y=train.iloc[:, 4]
test_X= test.iloc[:, 0:4]
test_y =test.iloc[:,4]

#model building & training
model = svm.SVC() #select algorithm
model.fit(train_X,train_y)  #model training
prediction=model.predict(test_X) #prediction
print('The accuracy of the SVM is:',metrics.accuracy_score(prediction,test_y))
```

Appendix 1.2 Practice: Use the train model to do prediciton

In [ ]:
```
print ("test data size :", test.shape)
test_X2 = test.iloc[0:5, 0:4]
test_y2 = test.iloc[0:5, 4]
print ("test data for input feature size: ", test_X2.shape)
p=model.predict(test_X2) #prediction
print(p)
df_compare = pd.DataFrame({'Predicted':p, 'Actual':test_y2})
print("\nCompare the predict and actual result: \n", df_compare)
```