Question : Describe your machine learning project in terms of the following:

What problem were you trying to solve?

What data did you collect?

What classes did you use to classify your data, and why they were chosen?

Solution

I am trying to solve IRIS DATA-SET (It is the most common problem in Machine Learning)

```
        Overview :
```

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals (in centimeters)

Column name : Sepal Length , Sepal Width , Petal Length , Petal Width

Based on the combination of these four features:

1. I am trying to predict Class of Flower (Train Model to distinguish the species from each other.)
1. It is a multivariate data set ie (variation of Iris flowers of three related species) It can be found on UCI's Machine Learning Reprository
1. class 0: Iris setosa (ie , Setosa)

   class 1: Iris virginica(ie, Virginica)

   class 2: Iris versicolor (ie, Versicolor)

Importing and loading Preprocessed Dataset

In [1]:
```
from sklearn.datasets import load_iris
df = load_iris()
```

In [2]:
```
print("Type of data: {}".format(type(df['data'])))
```

```
Type of data: <class 'numpy.ndarray'>
```

In [3]:
```
print("Shape of data: {}".format(df['data'].shape))
```

```
Shape of data: (150, 4)
```

In [5]:
```
print("First five columns : \n {}".format(df['data'][:5]))
```

```
First five columns :
 [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
```

In [6]:
```
print("Type of target: {}".format(type(df['target'])))
```

```
Type of target: <class 'numpy.ndarray'>
```

In [7]:
```
print("Keys of iris dataset: {}".format(df.keys()))
```

```
Keys of iris dataset: dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names',
'filename'])
```

In [8]:
```
print("Target names: {}".format(df['target_names']))
```

```
Target names: ['setosa' 'versicolor' 'virginica']
```

In [9]:
```
print("Feature names: {}".format(df['feature_names']))
```

```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (c
m)']
```

In [10]:
```
print("Shape of Target: {}".format(df['target'].shape))
```

```
Shape of Target: (150,)
```

In [11]:
```
print('Target: \n{}'.format(df['target']))
```

```
Target:
[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 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 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 2]
```

In [12]:
```
X = df['data']
y = df['target']
```

Train test split --- 75 % and 25% (default)

In [13]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
```

In [14]:
```
print('X train shape: {}'.format(X_train.shape))
print('X test shape: {}'.format(X_test.shape))
```

```
X train shape: (112, 4)
X test shape: (38, 4)
```

In [15]:
```
print('y train shape: {}'.format(y_train.shape))
print('y test shape: {}'.format(y_test.shape))
```

```
y train shape: (112,)
y test shape: (38,)
```

In [22]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

In [23]:
```
knn.fit(X_train,y_train)
```

Out[23]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

In [24]:
```
y_pred = knn.predict(X_test)
```

In [25]:
```
print("Test set prediction: {}".format(y_pred))
```

```
Test set prediction: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1
0
 2]
```

In [26]:
```
print("Score: {}".format(knn.score(X_test,y_test)))
```

```
Score: 0.9736842105263158
```

In [ ]:
```

```

Train Test Split --- 70% and 30 %

In [27]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,random_state = 0)
```

In [28]:
```
print('X train shape: {}'.format(X_train.shape))
print('X test shape: {}'.format(X_test.shape))
```

```
X train shape: (105, 4)
X test shape: (45, 4)
```

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

In [30]:
```
knn.fit(X_train,y_train)
```

Out[30]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

In [31]:
```
y_pred = knn.predict(X_test)
print("Test set prediction: {}".format(y_pred))
```

```
Test set prediction: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1
0
 2 1 1 2 0 2 0 0]
```

In [32]:
```
print("Score: {}".format(knn.score(X_test,y_test)))
```

```
Score: 0.9777777777777777
```

In [ ]:
```

```

In [33]:
```
knn5 = KNeighborsClassifier(n_neighbors=5)
knn5.fit(X_train,y_train)
```

Out[33]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [34]:
```
y_pred5 = knn5.predict(X_test)
print("Test set prediction: {}".format(y_pred5))
```

```
Test set prediction: [2 1 0 2 0 2 0 1 1 1 2 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1
0
 2 1 1 2 0 2 0 0]
```

In [35]:
```
print("Score: {}".format(knn5.score(X_test,y_test)))
```

```
Score: 0.9777777777777777
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

Question :

This part of the question is about evaluating your project, in terms of the following:

1. How well did your classifier work?
2. Were you happy with the results?
3. What was the overall percentage accuracy on test data?
4. Describe what types of images your models classifies well and which they classify badly. Explain why you think it performed well or badly on the images you described in the last part. Were there problems with the classifier that you were able to solve?

Describe your strategy for solving the problem

Solution :

Overview:

I used K-Nearest Neighbor(KNN) classification algorithm to predict the output. Libraries used:

- numpy
- pandas
- matplotlib.pyplot
- sklearn

1. Classifier works quite well, Yeah , I am happy with the results as the overall accuracy was 0.97777 (97.77%)
2. The model classify Setosa well , but faces some problem in other two classes(Virginica , Versicolor)
3. It performes well because it gives 97% accuracy on test set .
4. At the begining I use n_neighbors = 1 ie (the value of knn) , test set =30% , training set=70% :

```
                                                The model give Accuracy of 0.97777 (97.77%)
                                        n_neighbors = 5 ie (the value of knn) on the same te
    st and training set :
                                                  The model give Accuracy of 0.97777 (97.77%)
```

   but on increasing test to 40% and decresaing training set to 60% :

```
      The accuracy were as follows :
                        knn = 1 -------------- accuracy :  0.95555 (95.55%)
                        knn = 5 -------------- accuracy : 0.966667 (96.67%)
```