Pattern Recognition Course Project

Mina Sherif Henry

Environment

Language: Python 3.6

IDE: Pycharm

Machine Learning Library: Scikit-learn

Github Link: https://github.com/mina37/PatternRecognition

Dataset

Titanic Dataset

https://www.kaggle.com/c/titanic/data

Classification

Based on the data 4 features were used to teach four different algorithms and make them after that predict if a different Titanic passenger has survived or not.

The features being:

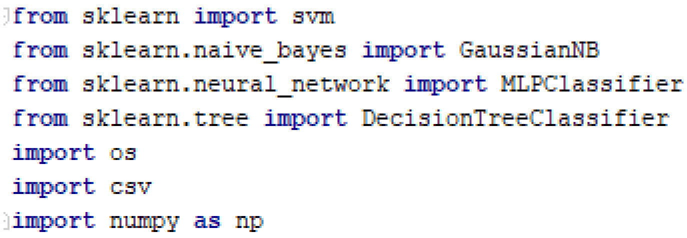
Sex(Gender), Age, SibSp(Sibling or Spouse) and Parch(Parents or Children)

Also to give everything a numerical value the Gender what given 0 for male and 1 for female. Any missing age data for any passenger was automatically replaced by 25. However at this point if all the available ages were averaged will give better results probably.

Classification algorithms used: **SVM, Naïve Bayes, Multi-layer Neural Network, Decision Tree.**

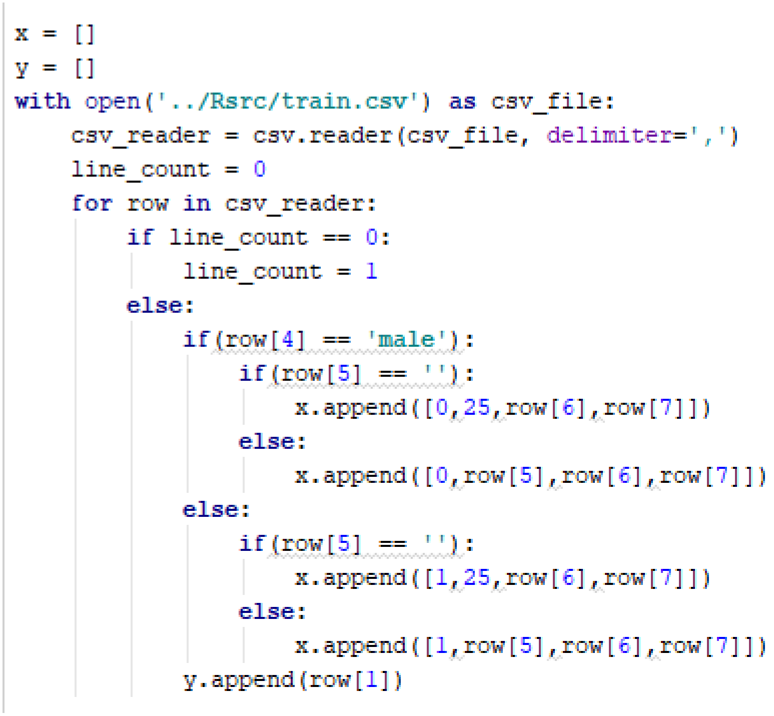
Code and Results

The code uses 2 main libraries: Scikit-learn and Numpy



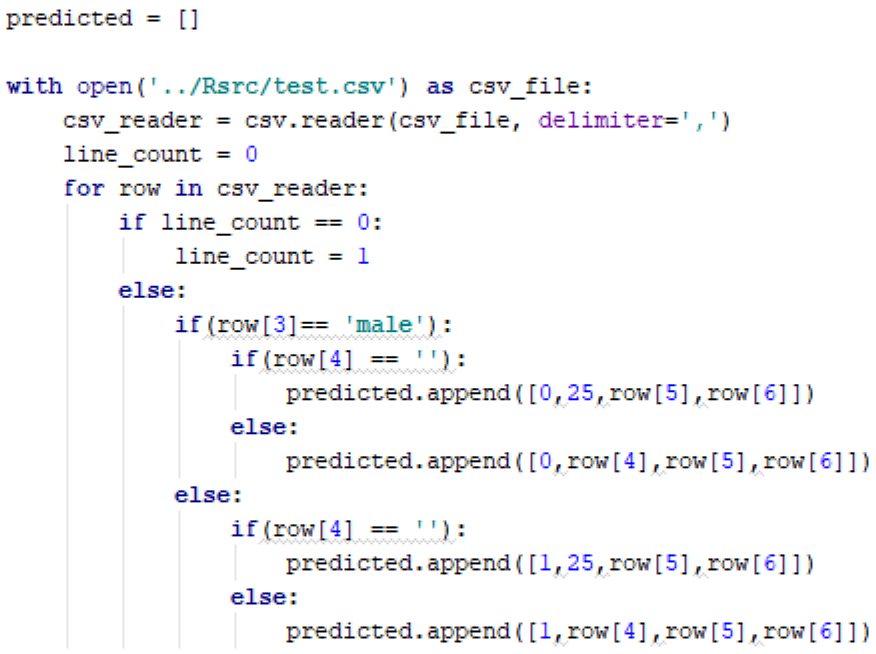
```
from sklearn import svm
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
import os
import csv
import numpy as np
```

We read the features of the training data in a 2- dimensional array “x” and the class that these features belong to – Certain passenger being survived or not – in an array “y”



```
x = []
y = []
with open('../Rsrc/train.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            line_count = 1
        else:
            if(row[4] == 'male'):
                if(row[5] == ''):
                    x.append([0,25,row[6],row[7]])
                else:
                    x.append([0,row[5],row[6],row[7]])
            else:
                if(row[5] == ''):
                    x.append([1,25,row[6],row[7]])
                else:
                    x.append([1,row[5],row[6],row[7]])
            y.append(row[1])
```

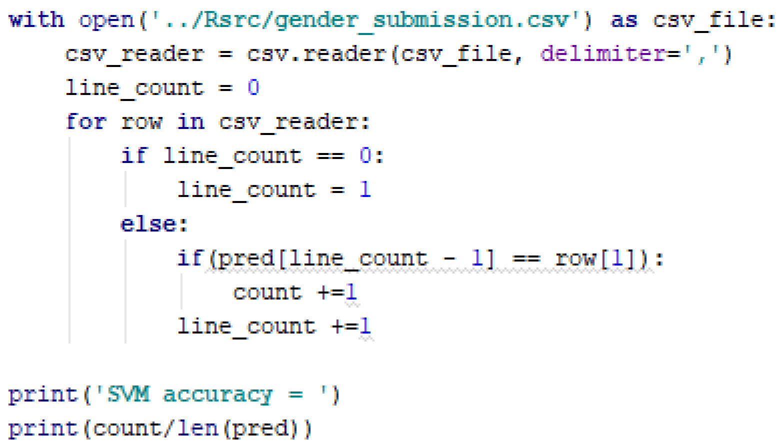
An array “Predicted” was filled with features of the test data



```
predicted = []

with open('../Rsrc/test.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            line_count = 1
        else:
            if(row[3]== 'male'):
                if(row[4] == ''):
                    predicted.append([0,25,row[5],row[6]])
                else:
                    predicted.append([0,row[4],row[5],row[6]])
            else:
                if(row[4] == ''):
                    predicted.append([1,25,row[5],row[6]])
                else:
                    predicted.append([1,row[4],row[5],row[6]])
```

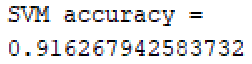
SVM

SVM was fitted with default configuration. And Tested on the Test Data



```
with open('../Rsrc/gender_submission.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            line_count = 1
        else:
            if(pred[line_count - 1] == row[1]):
                count +=1
            line_count +=1

print('SVM accuracy = ')
print(count/len(pred))
```

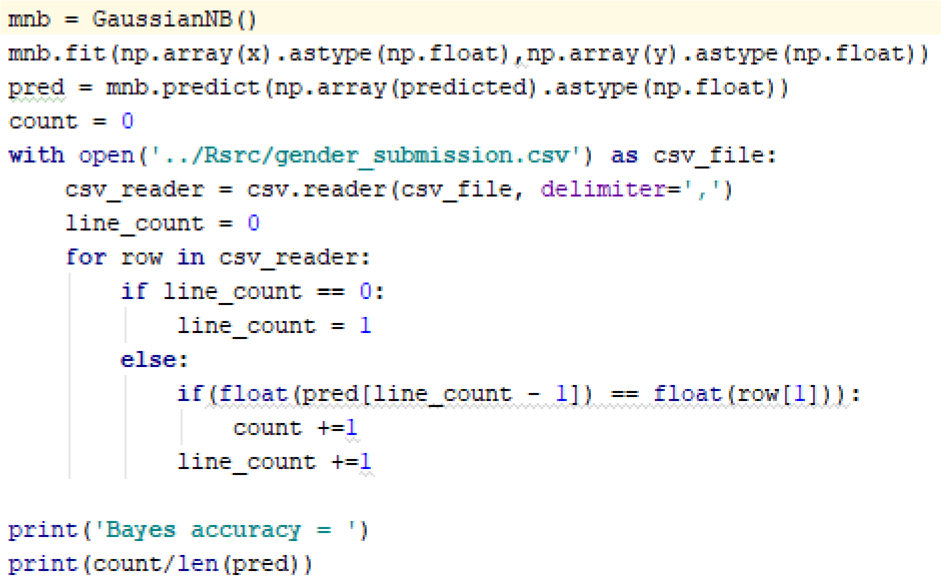
And acquired results of 91%



```
SVM accuracy =
0.916267942583732
```

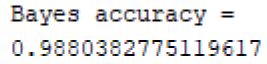
Bayes

Then Gaussian Naïve Bayes was fitted with the training data. Also with default configurations and was tested with the test data



```
mnb = GaussianNB()
mnb.fit(np.array(x).astype(np.float),np.array(y).astype(np.float))
pred = mnb.predict(np.array(predicted).astype(np.float))
count = 0
with open('../Rsrc/gender_submission.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            line_count = 1
        else:
            if(float(pred[line_count - 1]) == float(row[1])):
                count +=1
            line_count +=1

print('Bayes accuracy = ')
print(count/len(pred))
```

Got an accuracy of 98.8%

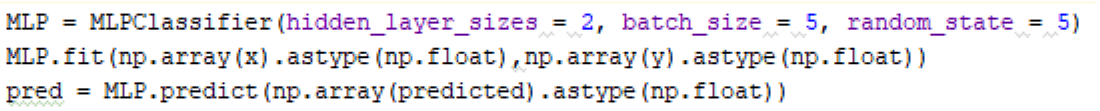


```
Bayes accuracy =
0.9880382775119617
```

Neural Network

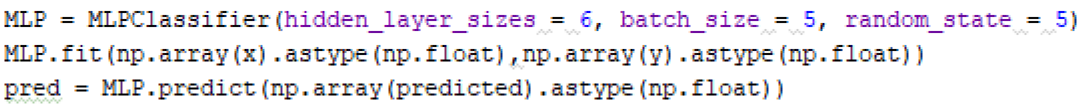
Multi-Layer neural network was fitted with the data. However for the sake of testing it, we decided to try different number of perceptrons in the hidden layers. In the model used the number of perceptrons affected the number of layers.

Hidden Layer Sizes (2)



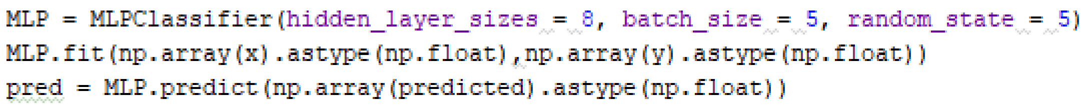
```
MLP = MLPClassifier(hidden_layer_sizes = 2, batch_size = 5, random_state = 5)
MLP.fit(np.array(x).astype(np.float),np.array(y).astype(np.float))
pred = MLP.predict(np.array(predicted).astype(np.float))
```

Hidden Layer Sizes (6)



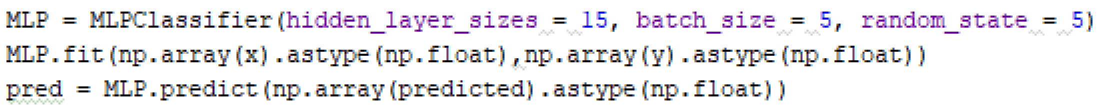
```
MLP = MLPClassifier(hidden_layer_sizes = 6, batch_size = 5, random_state = 5)
MLP.fit(np.array(x).astype(np.float),np.array(y).astype(np.float))
pred = MLP.predict(np.array(predicted).astype(np.float))
```

Hidden Layer Sizes (8)



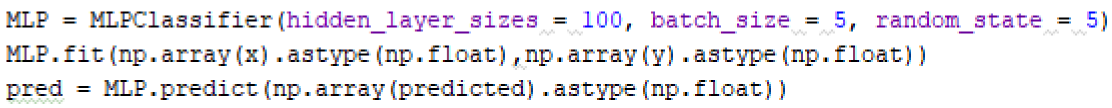
```
MLP = MLPClassifier(hidden_layer_sizes = 8, batch_size = 5, random_state = 5)
MLP.fit(np.array(x).astype(np.float),np.array(y).astype(np.float))
pred = MLP.predict(np.array(predicted).astype(np.float))
```

Hidden Layer Sizes (15)



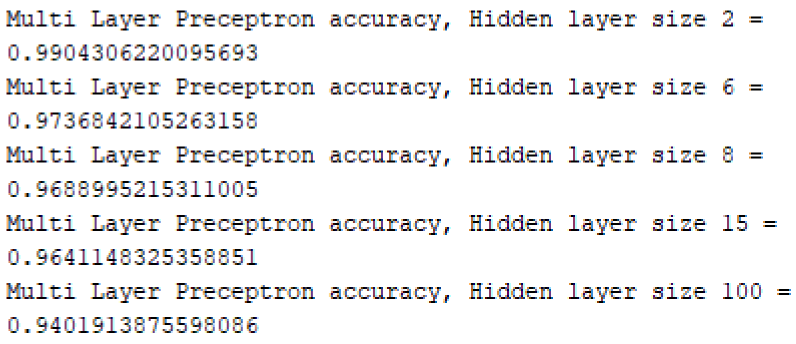
```
MLP = MLPClassifier(hidden_layer_sizes = 15, batch_size = 5, random_state = 5)
MLP.fit(np.array(x).astype(np.float),np.array(y).astype(np.float))
pred = MLP.predict(np.array(predicted).astype(np.float))
```

Hidden Layer Sizes (100)



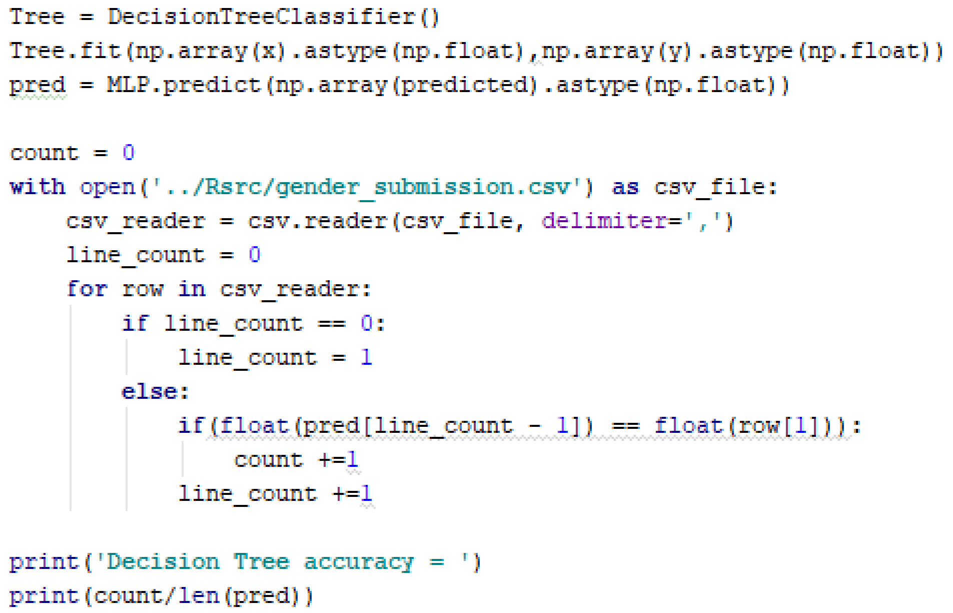
```
MLP = MLPClassifier(hidden_layer_sizes = 100, batch_size = 5, random_state = 5)
MLP.fit(np.array(x).astype(np.float),np.array(y).astype(np.float))
pred = MLP.predict(np.array(predicted).astype(np.float))
```

Results Acquired



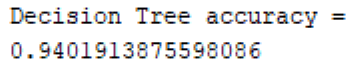
```
Multi Layer Preceptron accuracy, Hidden layer size 2 =
0.9904306220095693
Multi Layer Preceptron accuracy, Hidden layer size 6 =
0.9736842105263158
Multi Layer Preceptron accuracy, Hidden layer size 8 =
0.9688995215311005
Multi Layer Preceptron accuracy, Hidden layer size 15 =
0.9641148325358851
Multi Layer Preceptron accuracy, Hidden layer size 100 =
0.9401913875598086
```

Decision Tree



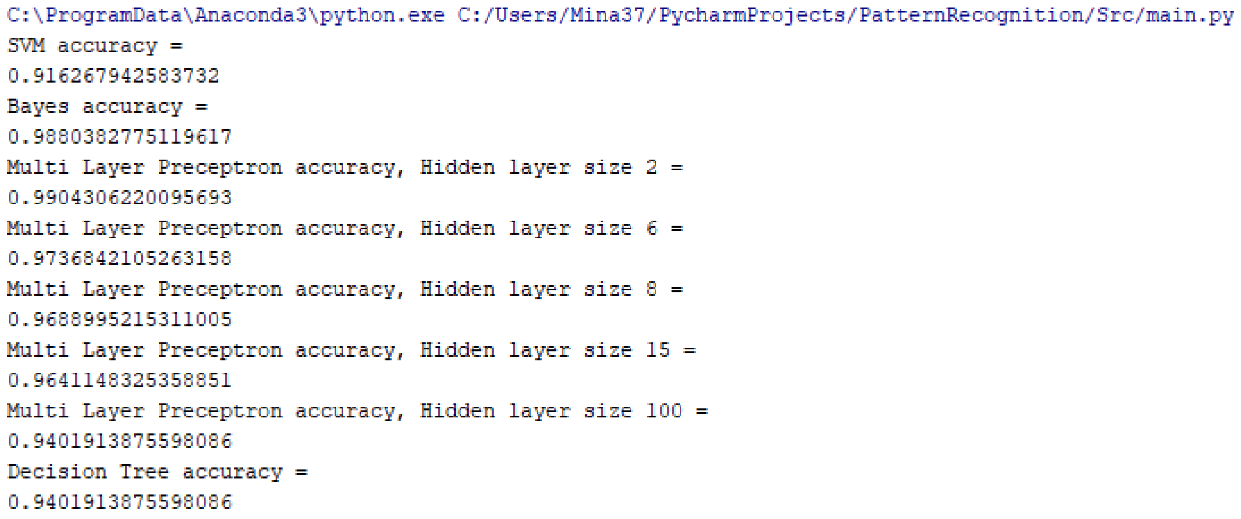
```
Tree = DecisionTreeClassifier()
Tree.fit(np.array(x).astype(np.float),np.array(y).astype(np.float))
pred = MLP.predict(np.array(predicted).astype(np.float))

count = 0
with open('../Rsrc/gender_submission.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            line_count = 1
        else:
            if(float(pred[line_count - 1]) == float(row[1])):
                count +=1
            line_count +=1

print('Decision Tree accuracy = ')
print(count/len(pred))
```

With results



```
Decision Tree accuracy =
0.9401913875598086
```

Overall Results and Conclusion



```
C:\ProgramData\Anaconda3\python.exe C:/Users/Mina37/PycharmProjects/PatternRecognition/Src/main.py
SVM accuracy =
0.916267942583732
Bayes accuracy =
0.9880382775119617
Multi Layer Preceptron accuracy, Hidden layer size 2 =
0.9904306220095693
Multi Layer Preceptron accuracy, Hidden layer size 6 =
0.9736842105263158
Multi Layer Preceptron accuracy, Hidden layer size 8 =
0.9688995215311005
Multi Layer Preceptron accuracy, Hidden layer size 15 =
0.9641148325358851
Multi Layer Preceptron accuracy, Hidden layer size 100 =
0.9401913875598086
Decision Tree accuracy =
0.9401913875598086
```

Conclusion:

The multi-layer neural network with the first configuration of 2 perceptrons in the hidden layers gave the best results of 99% accuracy