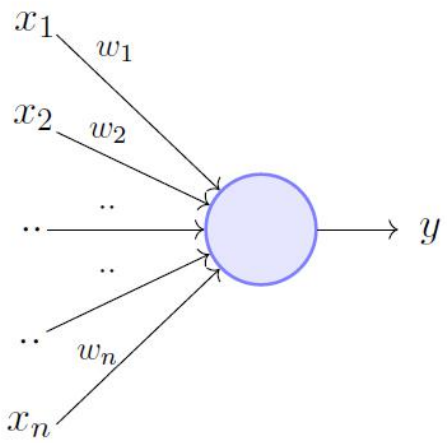
Machine Learning Lab 2

Perceptron Learning Algorithm

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. It takes an input, aggregates it (weighted sum) and returns 1 only if the aggregated sum is more than some threshold else returns 0. A single perceptron can only be used to implement linearly separable functions. It takes both real and boolean inputs and associates a set of weights to them, along with a bias (the threshold thing I mentioned above). We learn the weights, we get the function.



$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

The dataset

The dataset used to perform this experiment is the wine quality dataset, it is a combination of data on two types of wine variants, namely red wine and white wine, of the portuguese “Vinho Verde” wine. The dataset contains information on the parameters for fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

Experiment

In this experiment I have implemented a perceptron from scratch and and then used the matplotlib visualization library in python to plot the decision surface generated by the perceptron. This reveals how perceptron are very useful in cases where the data is linearly separable however they fail in the cases where is data is not linearly separable.

In this experiment, I have performed operations to clean the data, done statistical analysis of the data and then used various visualization tools to visualize the data in different ways which in turn reveals different information about the data which are otherwise not easily discernible.

Using the pandas library in I loaded the red wine and white wine datasets into the memory from their respective csv files and then merged the two datasets into one single pandas dataframe.

Using the pandas.Dataframe.describe() function in pandas I calculated the various statistical measures of each of the columns of the dataset.
* The dataset has a total of 4898 rows.
* The means for each of the columns are calculated as:
    Fixed acidity -> 6.85
    Volatile acidity -> 0.27
    Citric acid -> 0.33
    Residual sugar -> 6.39
    Chlorides -> 0.045
    Free sulphur dioxides -> 35.30
    Total sulphur dioxides -> 138.36
    Density -> 0.99
    pH -> 3.18
    Sulphates -> 0.48
    Alcohol -> 10.51

The pairplot function in seaborn library in python I was able to plot each column vs every other column in the dataset in the form of a scatter plot. And for also look at the target values for each data point in the form of a histogram.

Using the numpy library I implemented the Perceptron class along with the helper function for fitting a perceptron to a dataset and predicting the output from a data.

Initally I went ahead with fitting all the 11 parameters of the dataset but since plotting 11 dimensional data point is not possible, hence, for sake of visualization I chose the two most important features from the dataset fixed acidity and the volatile acidity.

From the visualization of the decision boundary it is clear that since the dataset was not linearly seperable the perceptron was not able to map a very clear boundary between the red and blue i.e the good and the bad wine samples. Examples can be seen with overlapping decision boundaries and thus one can conclude that the perceptron learning algorithm performs poorly on this task.

The code and plots can be found in the accompanying jupyter notebook.

Perceptron

November 1, 2018

Lab Assignment 2
Perceptron

0.0.1 Submitted to: Prof. Sweetlin Hemlatha

0.0.2 Submitted by: Prateek Singh(15BCE1091)

In [8]:
```
import numpy as np
import pandas as pd
import seaborn as sb

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

%matplotlib inline
```

Implementation of a perceptron from scratch

In [9]:
```
class Perceptron(object):
    """Perceptron classifier.

    Parameters
    ------------
    eta : float
        Learning rate (between 0.0 and 1.0)
    n_iter : int
        Passes over the training dataset.

    Attributes
    -----------
    w_ : 1d-array
        Weights after fitting.
    errors_ : list
        Number of misclassifications in every epoch.

    """
    def __init__(self, eta=0.03, n_iter=100):
        self.eta = eta
```

```
        self.n_iter = n_iter

    def fit(self, X, y):
        """Fit training data.

        Parameters
        ----------
        X : {array-like}, shape = [n_samples, n_features]
            Training vectors, where n_samples is the number of samples and
            n_features is the number of features.
        y : array-like, shape = [n_samples]
            Target values.

        Returns
        -------
        self : object

        """
        self.w_ = np.zeros(1 + X.shape[1])
        self.errors_ = []

        for _ in range(self.n_iter):
            errors = 0
            for xi, target in zip(X, y):
                update = self.eta * (target - self.predict(xi))
                self.w_[1:] += update * xi
                self.w_[0] += update
                errors += int(update != 0.0)
            self.errors_.append(errors)
        return self

    def net_input(self, X):
        """Calculate net input"""
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def activation(self, X):
        """Compute linear activation"""
        return self.net_input(X)

    def predict(self, X):
        """Return class label after unit step"""
        return np.where(self.activation(X) >= 0.0, 1, -1)


def plot_decision_regions(X, y, classifier, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
```

```
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # plot class samples
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=cmap(idx),
                    marker=markers[idx], label=cl)
```

```
df = pd.read_csv('../Dataset/winequality-red.csv', sep=';')
df2 = pd.read_csv('../Dataset/winequality-white.csv', sep=';')
df = pd.concat([df, df2])
df.tail()
```

Out[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
4893	6.2	0.21	0.29	1.6	0.039	
4894	6.6	0.32	0.36	8.0	0.047	
4895	6.5	0.24	0.19	1.2	0.041	
4896	5.5	0.29	0.30	1.1	0.022	
4897	6.0	0.21	0.38	0.8	0.020	

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
4893	24.0	92.0	0.99114	3.27	0.50	
4894	57.0	168.0	0.99490	3.15	0.46	
4895	30.0	111.0	0.99254	2.99	0.46	
4896	20.0	110.0	0.98869	3.34	0.38	
4897	22.0	98.0	0.98941	3.26	0.32	

	alcohol	quality
4893	11.2	6
4894	9.6	5
4895	9.4	6
4896	12.8	7
4897	11.8	6

```
In [10]: df.describe()
```

Out[10]:

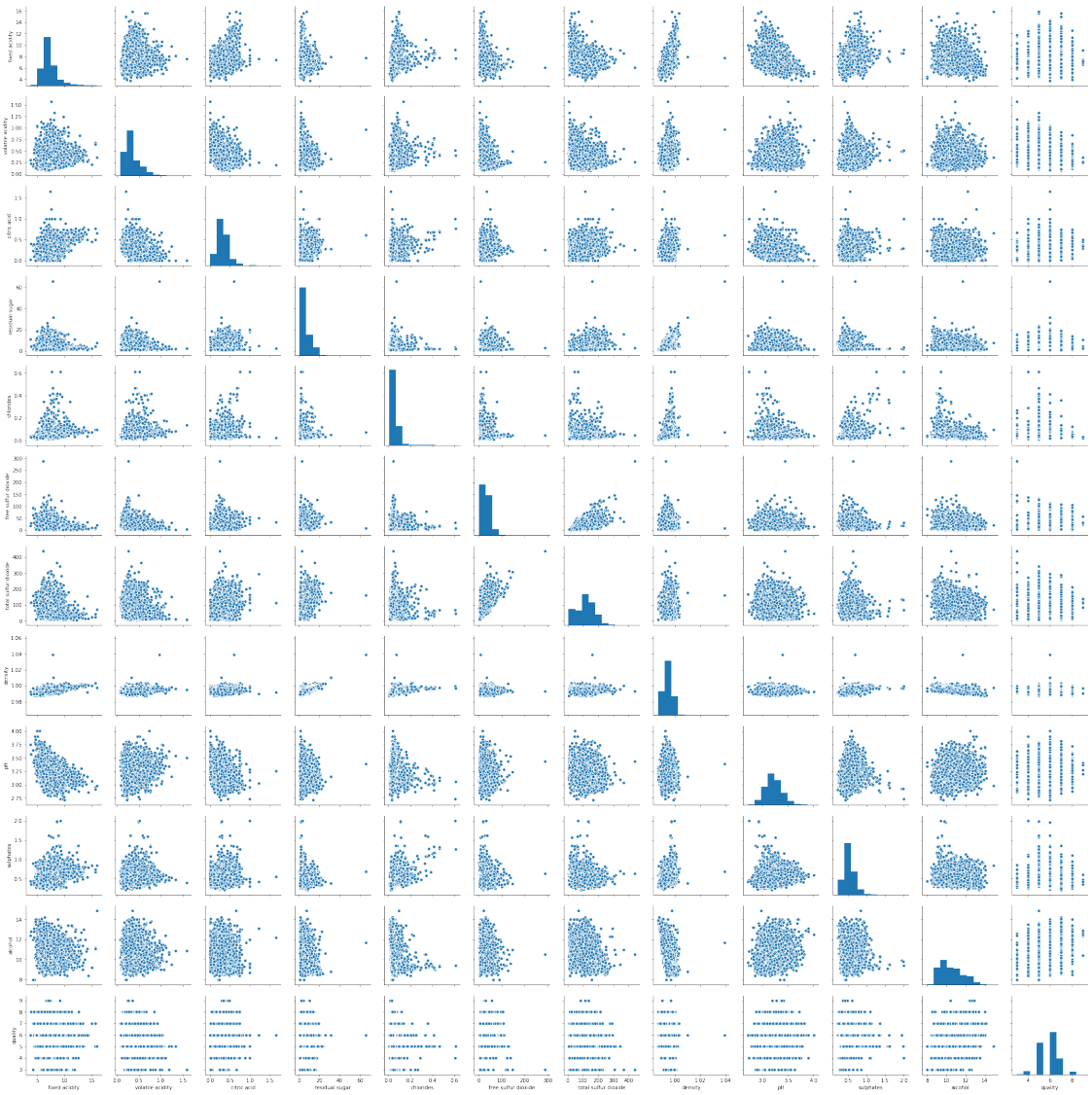
	fixed acidity	volatile acidity	citric acid	residual sugar
count	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235
std	1.296434	0.164636	0.145318	4.757804
min	3.800000	0.080000	0.000000	0.600000
25%	6.400000	0.230000	0.250000	1.800000
50%	7.000000	0.290000	0.310000	3.000000
75%	7.700000	0.400000	0.390000	8.100000
max	15.900000	1.580000	1.660000	65.800000

	chlorides	free sulfur dioxide	total sulfur dioxide	density
count	6497.000000	6497.000000	6497.000000	6497.000000
mean	0.056034	30.525319	115.744574	0.994697
std	0.035034	17.749400	56.521855	0.002999
min	0.009000	1.000000	6.000000	0.987110
25%	0.038000	17.000000	77.000000	0.992340
50%	0.047000	29.000000	118.000000	0.994890
75%	0.065000	41.000000	156.000000	0.996990
max	0.611000	289.000000	440.000000	1.038980

	pH	sulphates	alcohol	quality
count	6497.000000	6497.000000	6497.000000	6497.000000
mean	3.218501	0.531268	10.491801	5.818378
std	0.160787	0.148806	1.192712	0.873255
min	2.720000	0.220000	8.000000	3.000000
25%	3.110000	0.430000	9.500000	5.000000
50%	3.210000	0.510000	10.300000	6.000000
75%	3.320000	0.600000	11.300000	6.000000
max	4.010000	2.000000	14.900000	9.000000

In [11]:
```
sb.pairplot(df.dropna(), size=2.5)
```

Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x7f45705169e8>
```

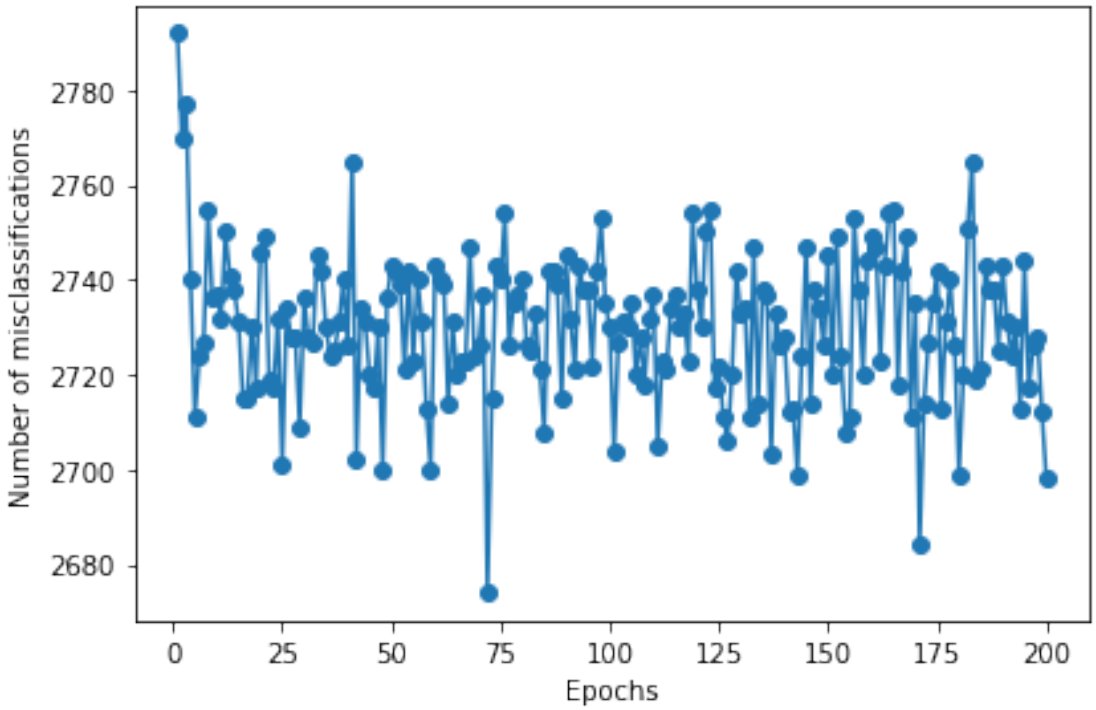


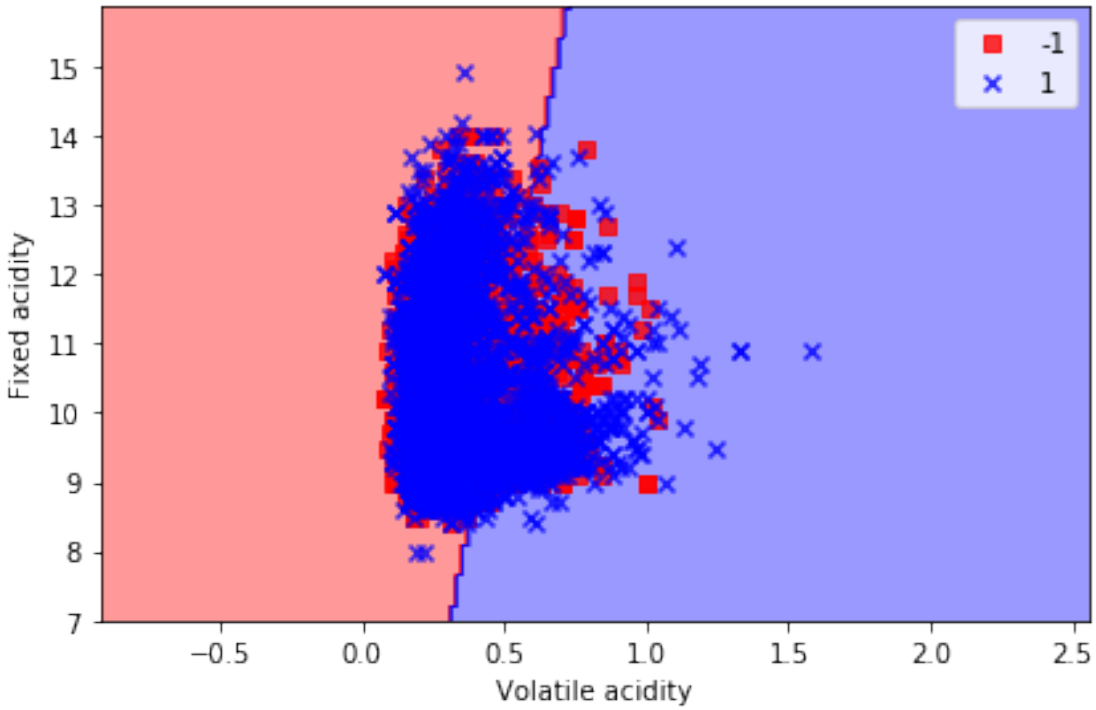
Fitting the perceptron with all the 11 features in the dataset

```
In [12]: y = df.iloc[:, 11].values
         y = np.where(y == 6, -1, 1)


         X = df.iloc[:, :11].values

         ppn = Perceptron(eta=0.001, n_iter=200)

         ppn.fit(X.astype(float), y)

Out[12]: <__main__.Perceptron at 0x7f455da21b70>
```

Fitting the perceptron with the 2 most important features in the dataset for visualization purposes

```
In [13]: y = df.iloc[:, 11].values
         y = np.where(y == 6, -1, 1)


         X = df.iloc[:, [1,10]].values

         ppn = Perceptron(eta=0.001, n_iter=200)

         ppn.fit(X.astype(float), y)

         plt.plot(range(1, len(ppn.errors_) + 1), ppn.errors_, marker='o')
         plt.xlabel('Epochs')
         plt.ylabel('Number of misclassifications')

         plt.tight_layout()
         plt.show()

         plot_decision_regions(X.astype(float), y, classifier=ppn)
         plt.xlabel('Volatile acidity')
         plt.ylabel('Fixed acidity')
         plt.legend(loc='upper right')

         plt.tight_layout()
         plt.show()
```




-1
1
Fixed acidity
Volatile acidity
15
14
13
12
11
10
9
8
7
−0.5
0.0
0.5
1.0
1.5
2.0
2.5