**“OZONE LEVEL DETECTION”**

A PROJECT REPORT

***Submitted by***

SUDESHNA NATH (CSE 3C, 21)

Enrolment Id - 3312016009001706

AVIRUP KUNDU (CSE 3C, 22)

Enrolment Id - 3312016009001048

**Data Science & Data Analytics Lab (CS – 695A)**

Project Guide - Prof. Sankhadeep Chatterjee & Prof. Somarpita Dutta

B. Tech 3rd year

2019



University of Engineering & Management (UEM)

University Area, Plot No. III - B/5, New Town, Action Area - III, Kolkata, West Bengal 700156

**CERTIFICATE**

Certified that this project report “**Ozone Level Detection Dataset”** is the bonafide work of

1. Sudeshna Nath (Enrollment no. -3312016009001706)
2. Avirup Kundu (Enrollment no. -3312016009001048)

of B.Tech, CSE, who carried out the project work under our supervision.

………………………………..

………………………………..

**SIGNATURE**

**Examiner:**

**ACKNOWLEDGEMENT**

The completion of this project could not have been accomplished

without the support of our teachers and guide Prof. Sankhadeep Chatterjee & Prof. Somarpita Dutta. We are thankful to you for allowing us your time to research and write.

We are also very thankful to our respected teachers for their co-operation and suggestion regarding the project work.

Last but not the least we are very thankful to our HOD, Prof. Sukalyan Goswami for giving us an opportunity of doing such an interesting project work.

* **Sudeshna Nath**
* **Avirup Kundu**

**OVERVIEW**

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across.

As it is evident from the name, it gives the computer that which makes it more similar to humans: The ability to learn.

Machine learning is actively being used today, perhaps in many more places than one would expect.

The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

**TYPES OF LEARNING:**

1. Supervised Learning
2. Unsupervised Learning

**INTRODUCTION**

Data mining is concerned with locating hidden relationships present in commercial enterprise data presents groups to make predictions for eventual use.

Data mining has risen as a key commercial enterprise intelligence technology. The preference of data mining is to extract implicit, previously unexplored and doubtlessly beneficial (or actionable) styles from statistics.

Data mining encompass many up to date tactics along with classification (neural networks, k-nearest neighbour, naive Bayes classifier, and decision trees), clustering (density based clustering, k-means, hierarchical clustering), association (constraint-based association, multilevel association, multidimensional, one-dimensional).

Years of education display that data mining is a technique, and it’s helpful application calls for post processing (presentation, understanding ability, summary), facts pre-processing (cleaning, noise/outlier removal, dimensionality reduction) true knowledge of problem domains and domain facility.

All traditional algorithms are exaggerated to some amount by the class imbalance crisis. Also, the accurate choice of the metric (or consolidation of metrics) to assess – and ultimately improve, is essential for the accomplishment of a data mining effort in such areas, since most of the time improving one metric degrades others.

**ALGORITHMS**

**SUPPORT VECTOR MACHINE (SVM):**

A support-vector machine constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space.

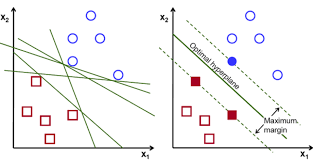
To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function k(x, y) selected to suit the problem.

The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vector is an orthogonal (and thus minimal) set of vectors that defines a hyperplane.

The vectors defining the hyperplanes can be chosen to be linear combinations with parameters {\displaystyle \alpha \_{i}}of images of feature vectors x that occur in the data base. With this choice of a hyperplane, the points {\displaystyle x}x in the feature space that are mapped into the hyperplane are defined by the relation.

Note that if k(x, y) becomes small as {\displaystyle y}y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding data base point. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated.

Note the fact that the set of points {\displaystyle x}x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets that are not convex at all in the original space.

{\displaystyle x\_{i}}{\displaystyle k(x,y)}{\displaystyle \textstyle \sum \_{i}\alpha \_{i}k(x\_{i},x)={\text{constant}}.}

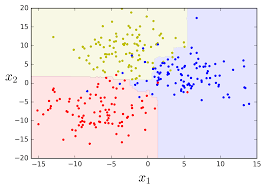
**RANDOM FOREST CLASSIFIER:**

Decision trees are a popular method for various machine learning tasks. Tree learning *"comes closest to meeting the requirements for serving as an off-the-shelf procedure for data mining*", because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspect able models.

However, they are seldom accurate. In particular, trees that are grown very deep tend to learn highly irregular patterns: they over fit their training sets, i.e. have low bias, but very high variance.

Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance.

This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.



**K- NEAREST NEIGHBOUR:**

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or Hamming distance).

In the context of gene expression microarray data, for example, *k*-NN has also been employed with correlation coefficients such as Pearson and Spearman.

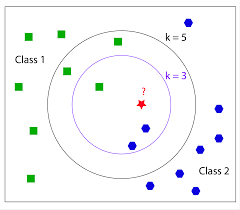
Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighborhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number.

One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its *k* nearest neighbors. The class (or value, in regression problems) of each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point.

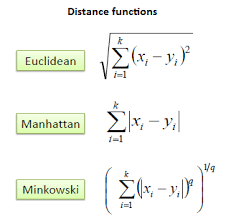
Another way to overcome skew is by abstraction in data representation.

For example, in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. *K*-NN can then be applied to the SOM.



**APPLICATION**

* SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labelled training instances in both the standard inductive and transductive settings. Some methods for shallow semantic parsing are based on support vector machines.
* Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.
* Hand-written characters can be recognized using SVM.
* The SVM algorithm has been widely applied in the biological and other sciences. They have been used to classify proteins with up to 90% of the compounds classified correctly. Permutation tests based on SVM weights have been suggested as a mechanism for interpretation of SVM models Support-vector machine weights have also been used to interpret SVM models in the past Posthoc interpretation of support-vector machine models in order to identify features used by the model to make predictions is a relatively new area of research with special significance in the biological sciences.



**SOURCE CODE**

1. SUPPORT VECTOR MACHINE

import matplotlib.pyplot as plt

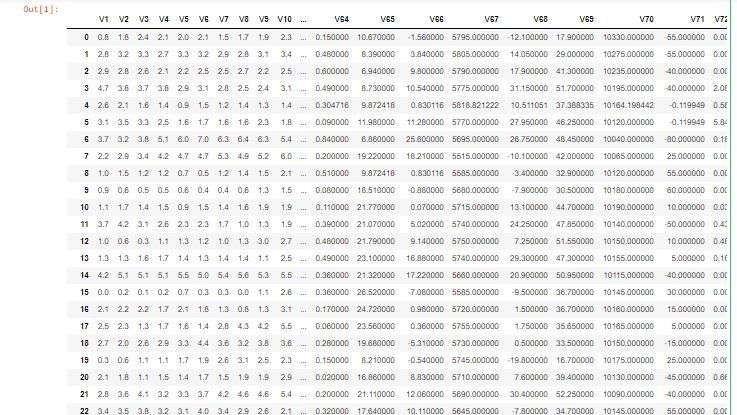
from copy import deepcopy

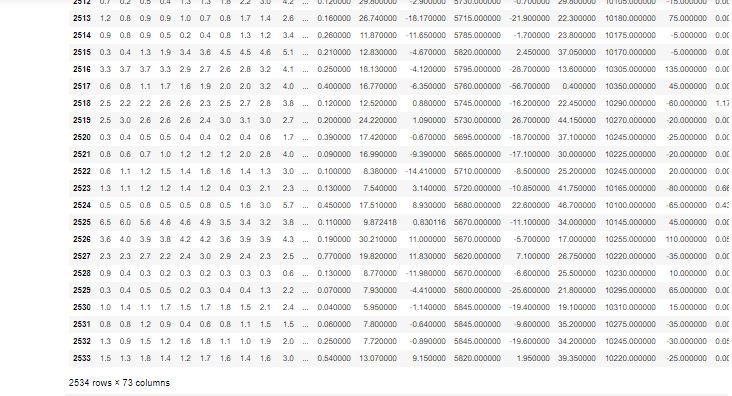
import numpy as np

import pandas as pd

data=pd.read\_csv("ozondataset.csv")

data.iloc[:,:]





x=data.iloc[:,1:72].values

y=data["Class"].values

print(y)

[1 1 1 ... 1 1 1]

from sklearn.model\_selection import train\_test\_split as tts

x\_train,x\_test,y\_train,y\_test= tts(x,y,test\_size=0.40,random\_state=42)

#support vector classifier

#test size+train size=1.then We have to train our machine than to test. so test\_size is <.5

#random\_state is used to pick trained data randomly

print(x\_train.shape)

print(x\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

(1520, 71)

(1014, 71)

(1520,)

(1014,)

from sklearn import svm#support vector machine

cl=svm.SVC(gamma=0.001)

cl.fit(x\_train,y\_train)

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape='ovr', degree=3, gamma=0.001, kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False)

y\_predict=cl.predict(x\_test)

from sklearn.metrics import accuracy\_score as acc

print(acc(y\_test,y\_predict))

0.9349112426035503

1. K NEAREST NEIGHBOUR

import matplotlib.pyplot as plt

import numpy as np

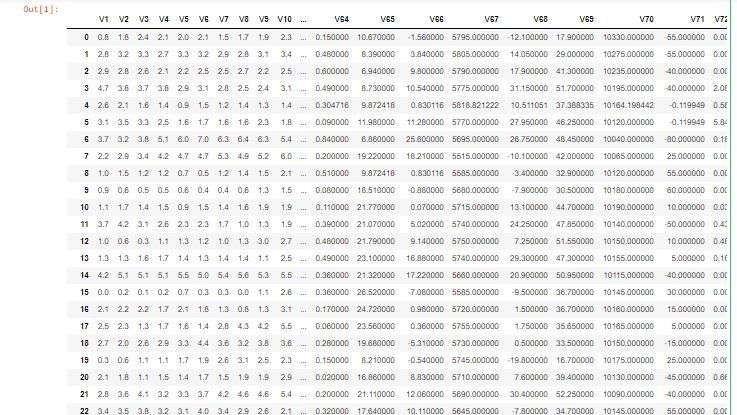
import pandas as pd

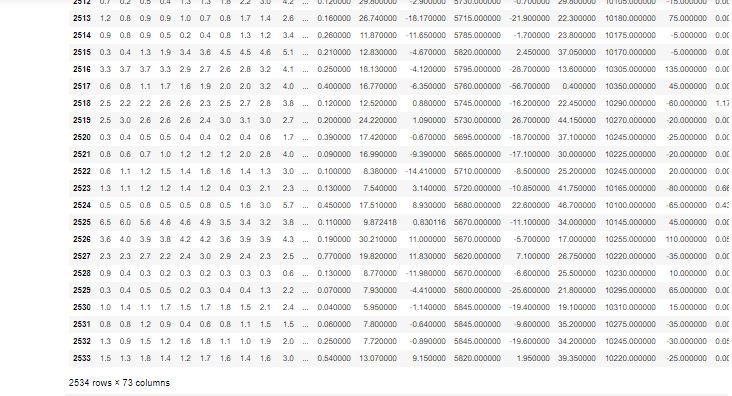
data = pd.read\_csv("ozondataset.csv")

print(data.shape)

data.iloc[:,:]

(2534, 73)





x1=data.iloc[:,1:72].values

y=data["Class"].values

print(y)

[1 1 1 ... 1 1 1]

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x1, y,test\_size=0.4)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=6)

knn.fit(x\_train, y\_train)

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=6, p=2,

weights='uniform')

y\_predict=knn.predict(x\_test)

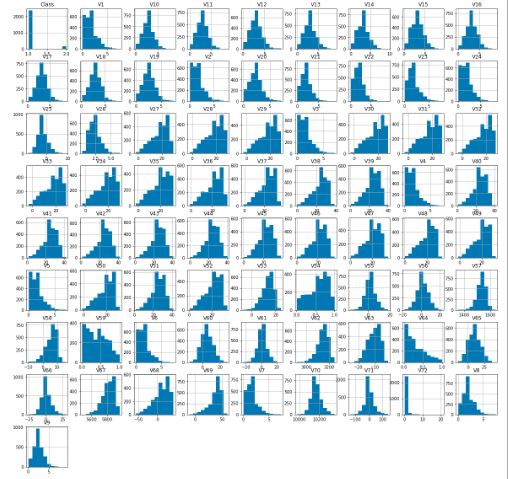
from sklearn.metrics import accuracy\_score as acc

print(acc(y\_test,y\_predict))

0.9447731755424064

data.hist(figsize = (20,20))

plt.show



1. RANDOM FOREST CLASSIFIER

import matplotlib.pyplot as plt

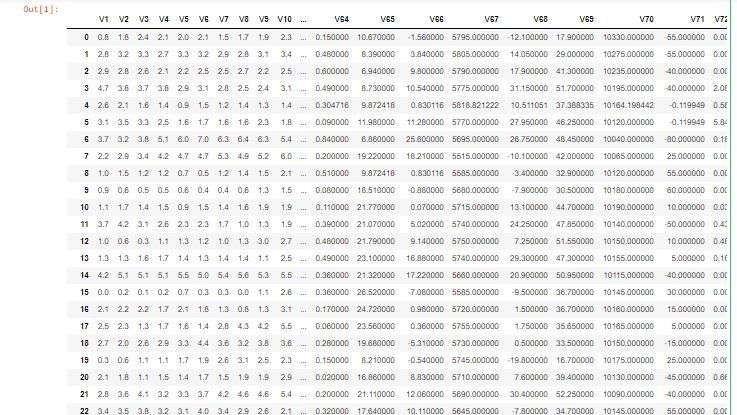
from copy import deepcopy

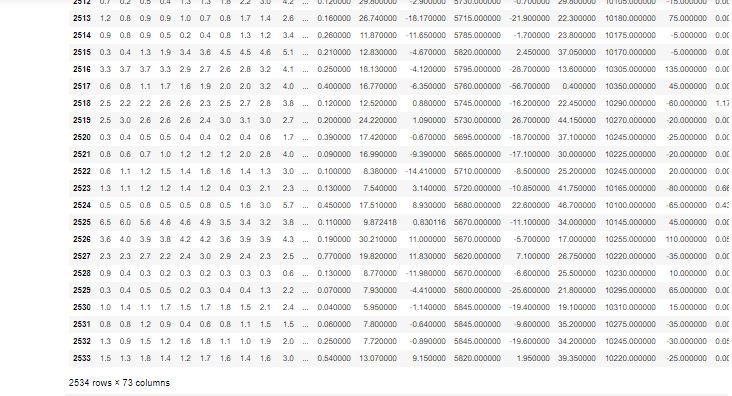
import numpy as np

import pandas as pd

data=pd.read\_csv("ozondataset.csv")

data.iloc[:,:]





x=data.iloc[:,1:72].values

y=data["Class"].values

print(y)

[1 1 1 ... 1 1 1]

from sklearn.model\_selection import train\_test\_split as tts

x\_train,x\_test,y\_train,y\_test= tts(x,y,test\_size=0.40,random\_state=42)

#support vector classifier

#test size+train size=1.then We have to train our machine than to test. so test\_size is <.5

#random\_state is used to pick trained data randomly

print(x\_train.shape)

print(x\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

(1520, 71)

(1014, 71)

(1520,)

(1014,)

from sklearn.ensemble import RandomForestClassifier

cl=RandomForestClassifier(n\_estimators=4)

cl.fit(x\_train,y\_train)

RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=4, n\_jobs=None,

oob\_score=False, random\_state=None, verbose=0,

warm\_start=False)

y\_predict=cl.predict(x\_test)

from sklearn.metrics import accuracy\_score as acc

print(acc(y\_test,y\_predict))

0.9398422090729783

**CONCLUSION**

After successful completion of this project we can conclude that Ozone Level Detection Dataset can be used in data science for good purpose.

Here we have used Support Vector Machine, Random Forest, K-Nearest Neighbour Classifier for analysis of Ozone Level Detection.

In case of SVM we got an accuracy of 0.9349112426035503

In case of KNN classifier we got an accuracy of 0.9447731755424064

In case of Random forest we got an accuracy of 0.9398422090729783